Rinascimento

Using Event-value Functions for Playing Splendor

Ivan Bravi
Simon Lucas

Queen Mary
University of London

Keywords

Splendor: the real board game*

ℜinascimento: the framework implementing the game(s)

Statistical Forward Planning (SFP): AI game-playing algorithms that use a forward-model to simulate future game states (e.g. MCTS, RHEA)

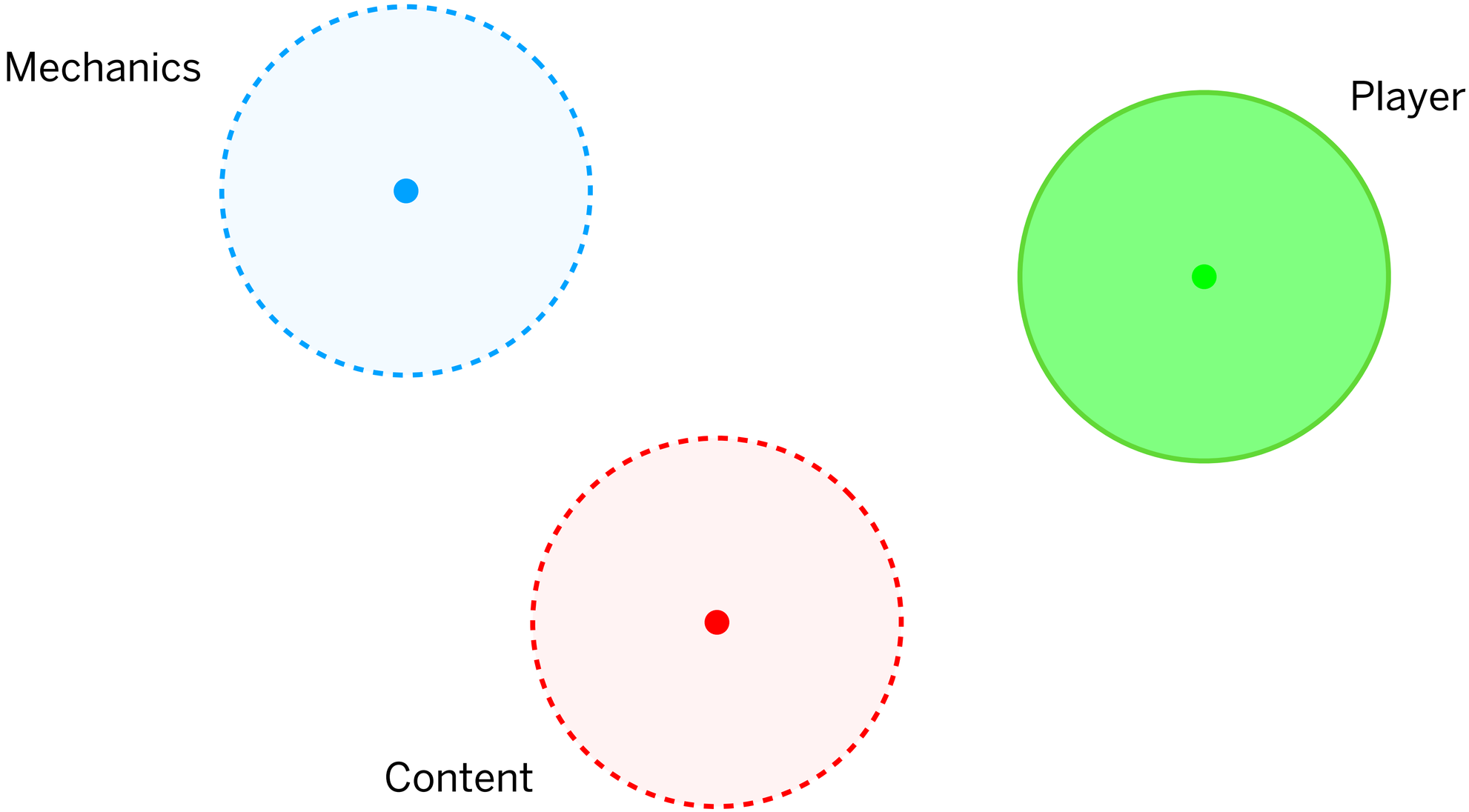
Hyperparameters: the parameters governing the behaviour of the algorithm

**designed by Marc André and published by Space Cowboys*

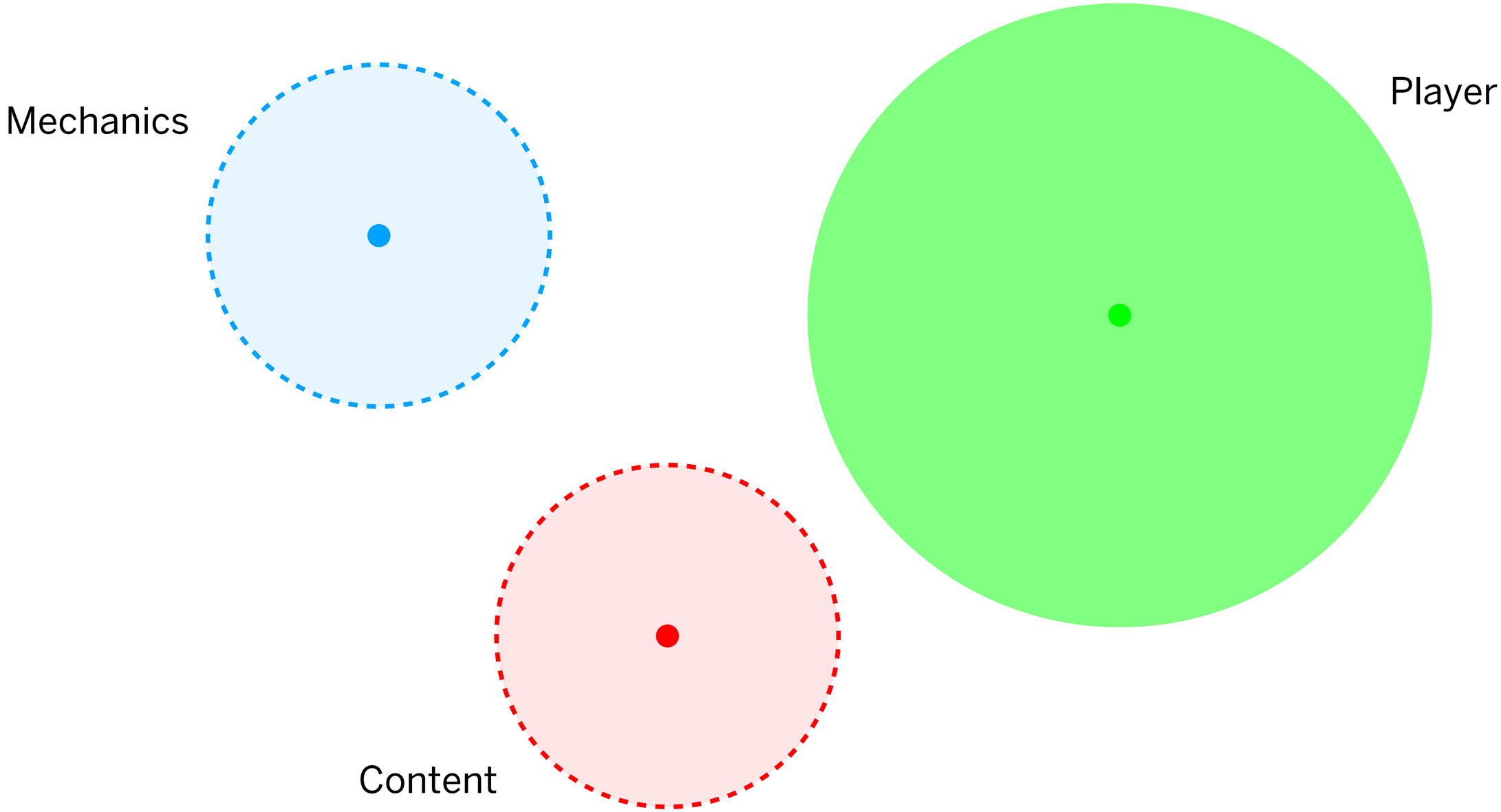
Preface

how I am approaching research

Rinascimento



Event-value function



Why more purpose?

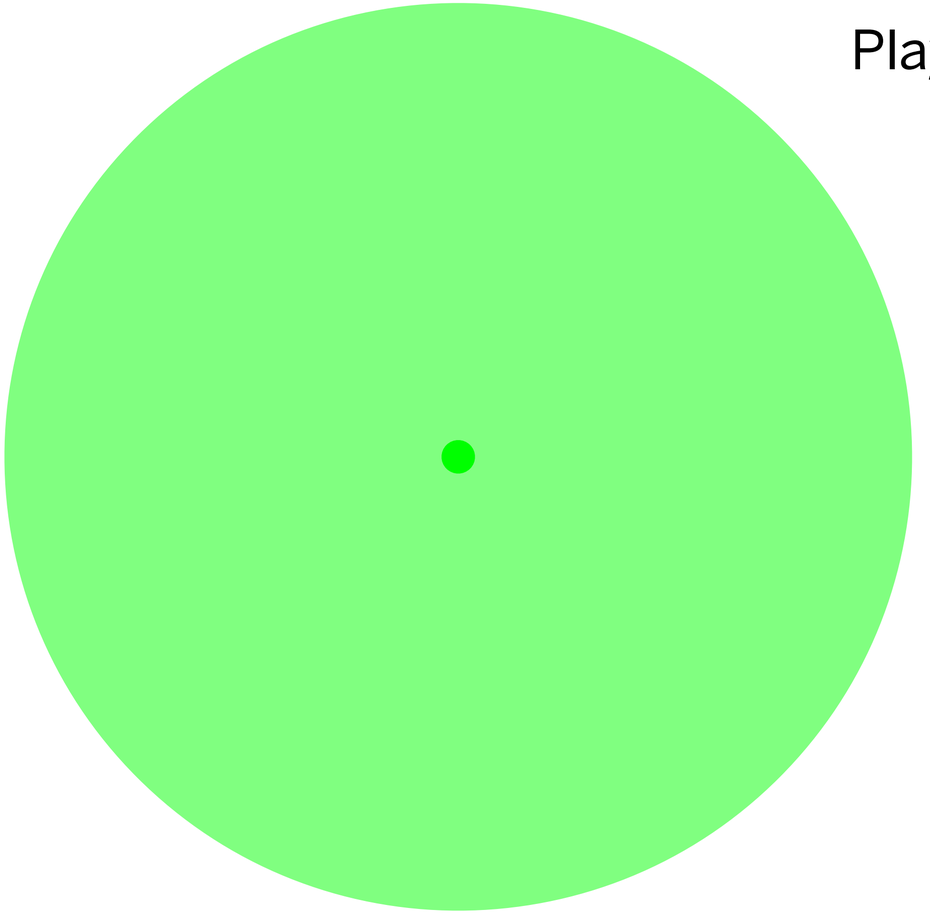
winning is for losers, have fun!

humans show amazingly different behaviours

AI shouldn't just ~~destroy~~ beat us all

We want more for AI

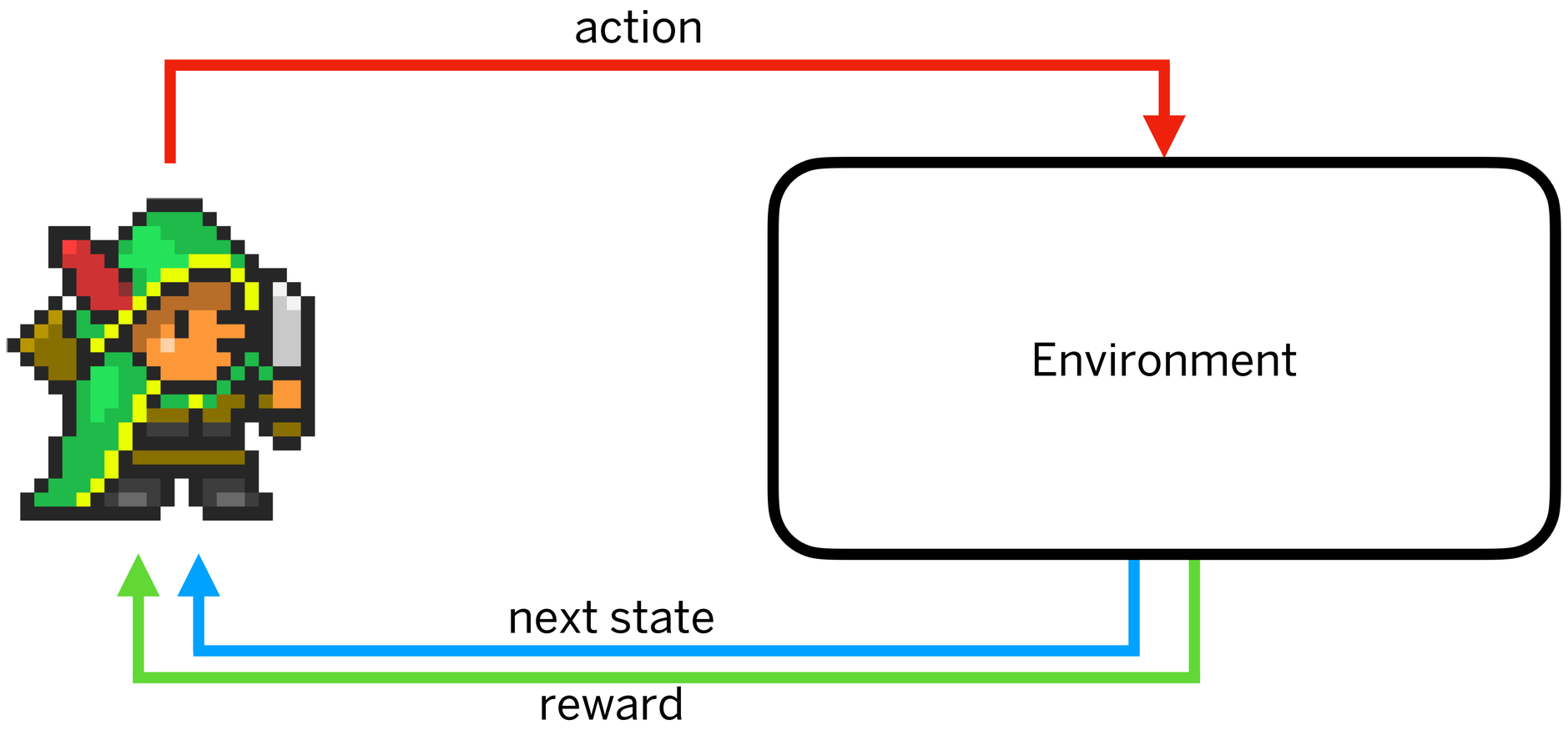
- more ~~conscious~~ aware of its decisions
- more flexible/varied behaviours
- more robust



Event-value Functions

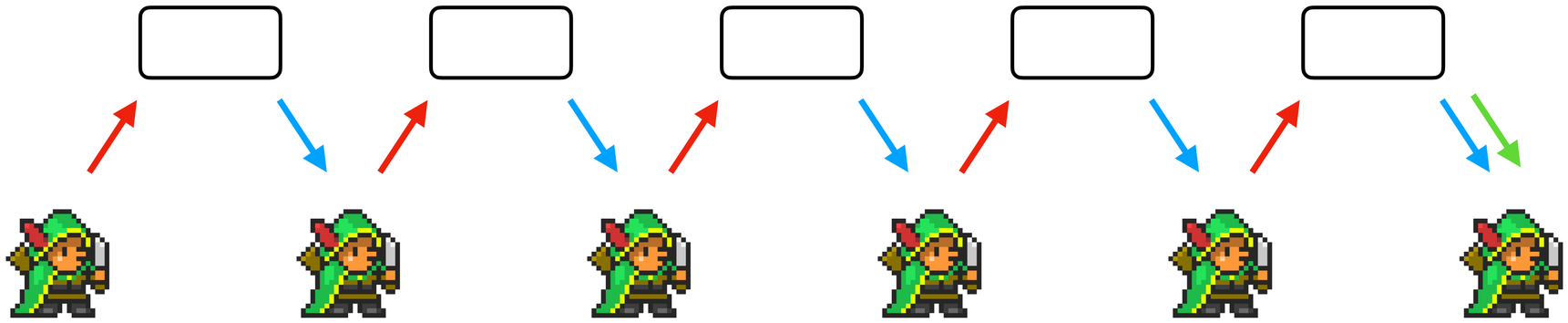
The novel bit

Agent - Environment



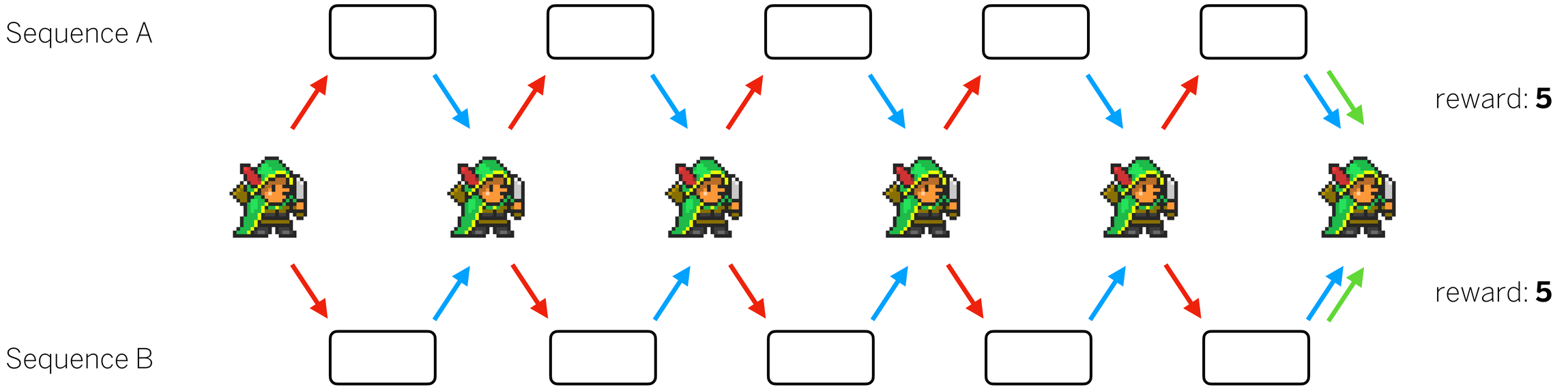
Agent - Environment

Sequence A

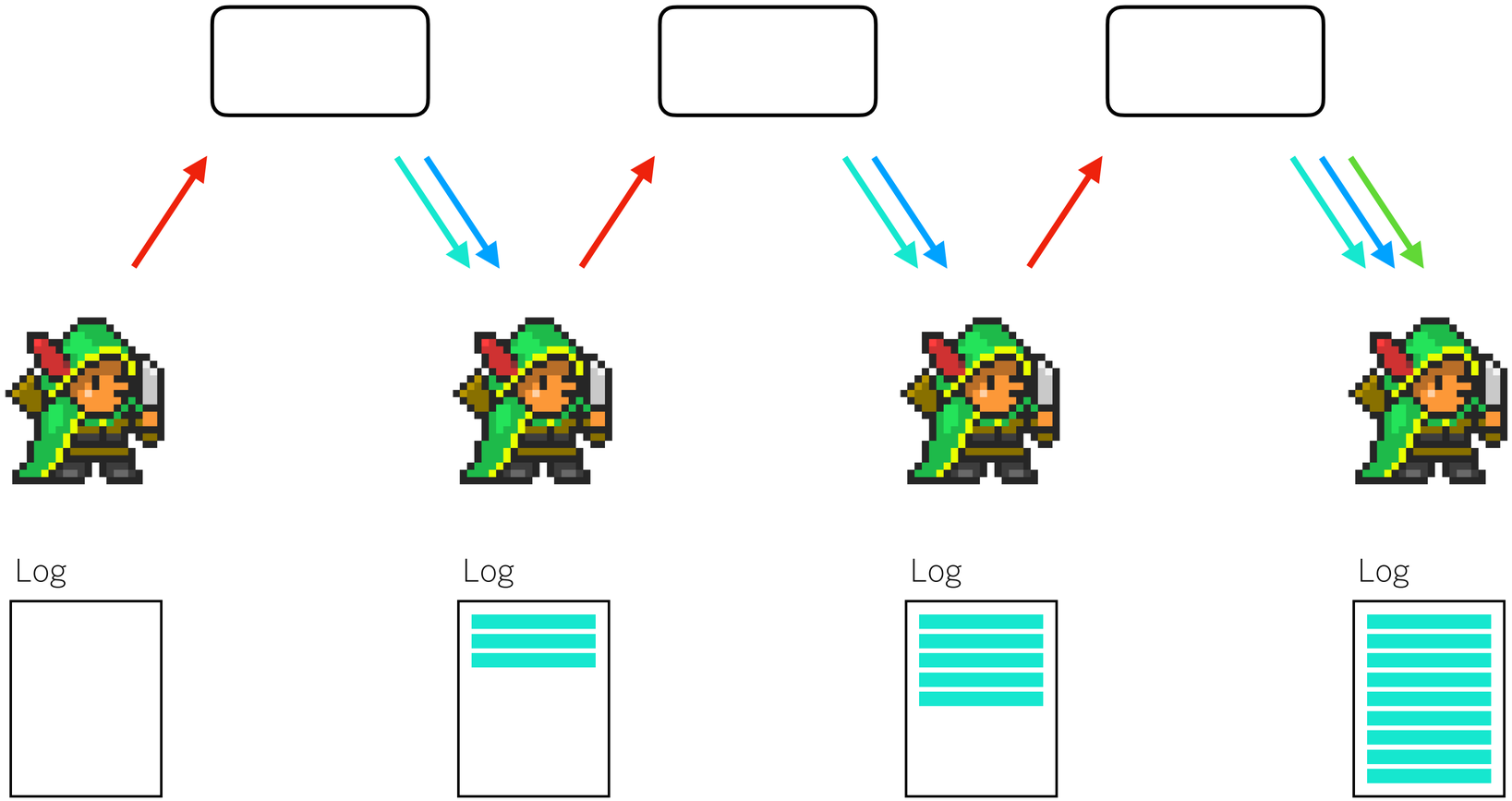


reward: **5**

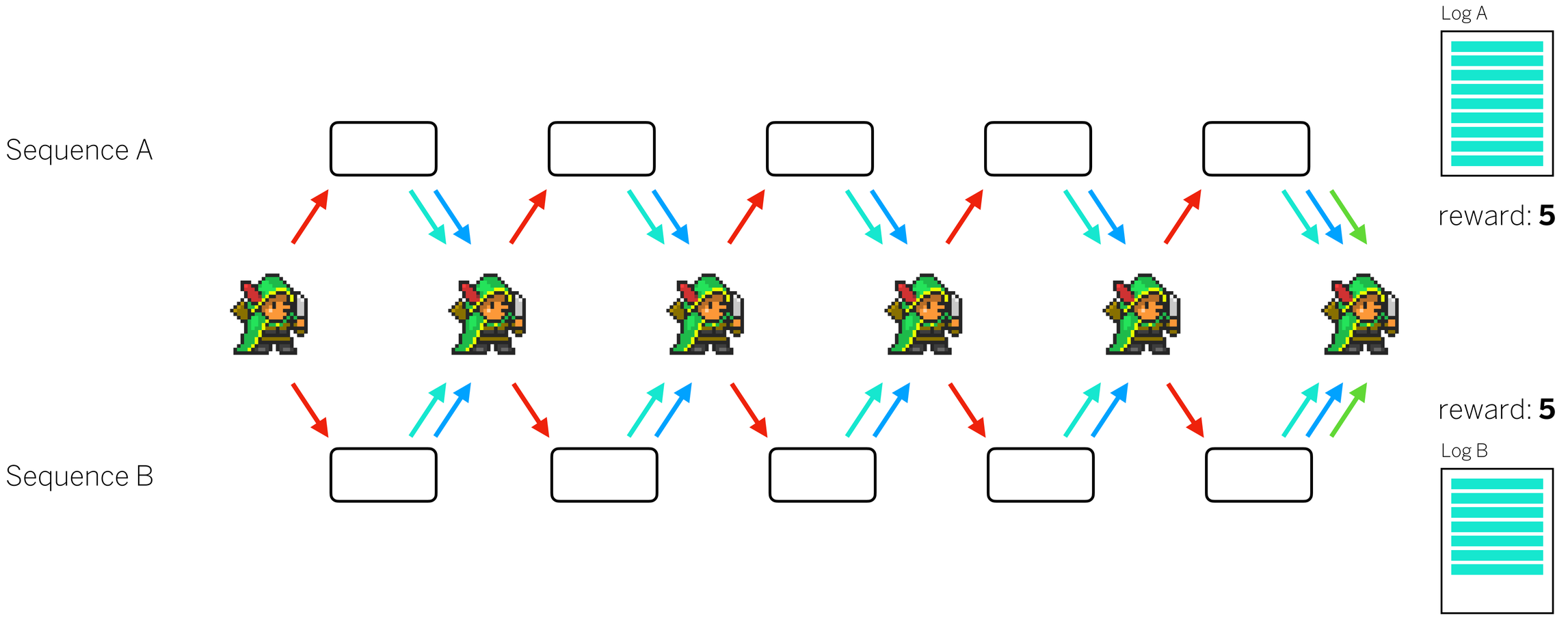
Agent - Environment



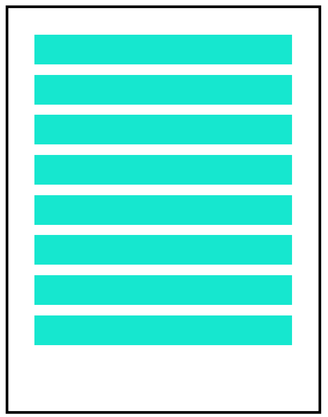
Agent - Environment - Log

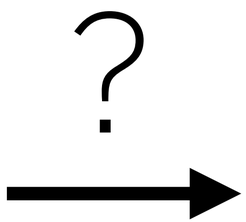


Agent - Environment - Log



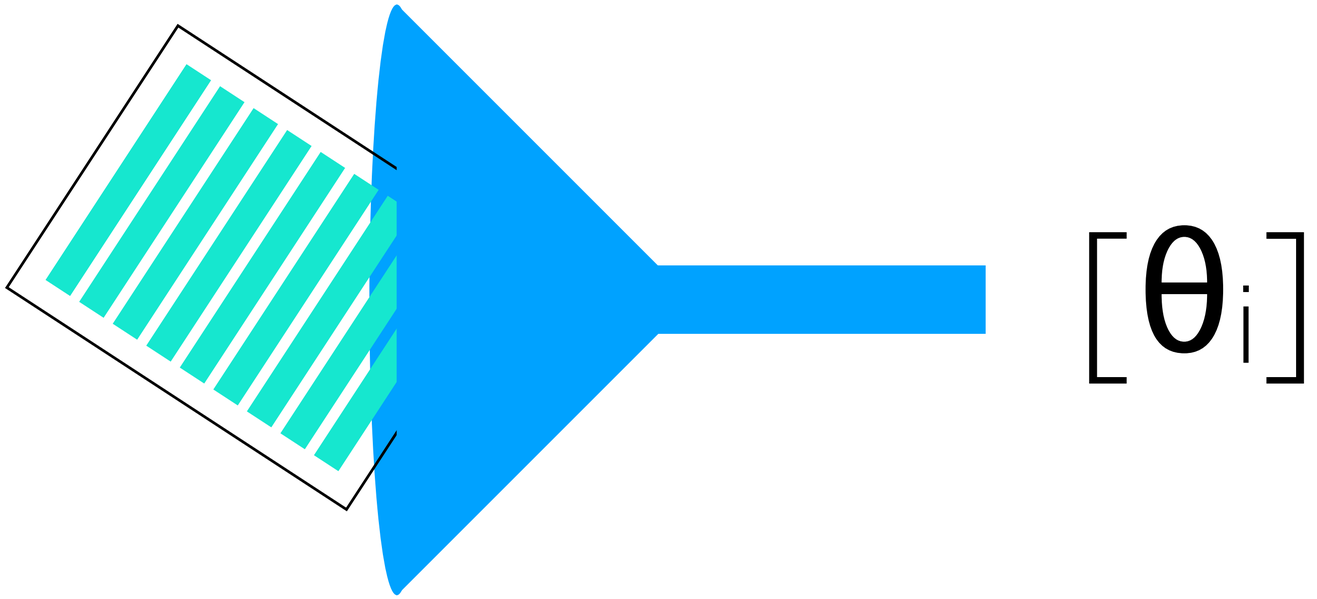
What now?





42

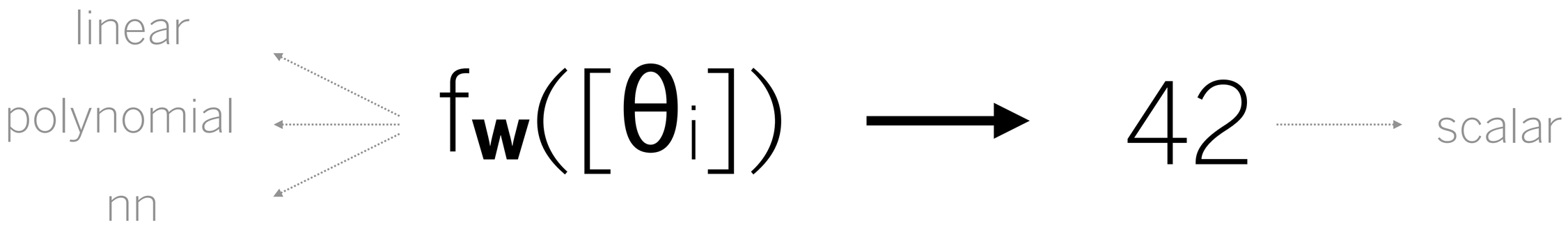
Synthesise



synthesise a log in a fixed-size features vector

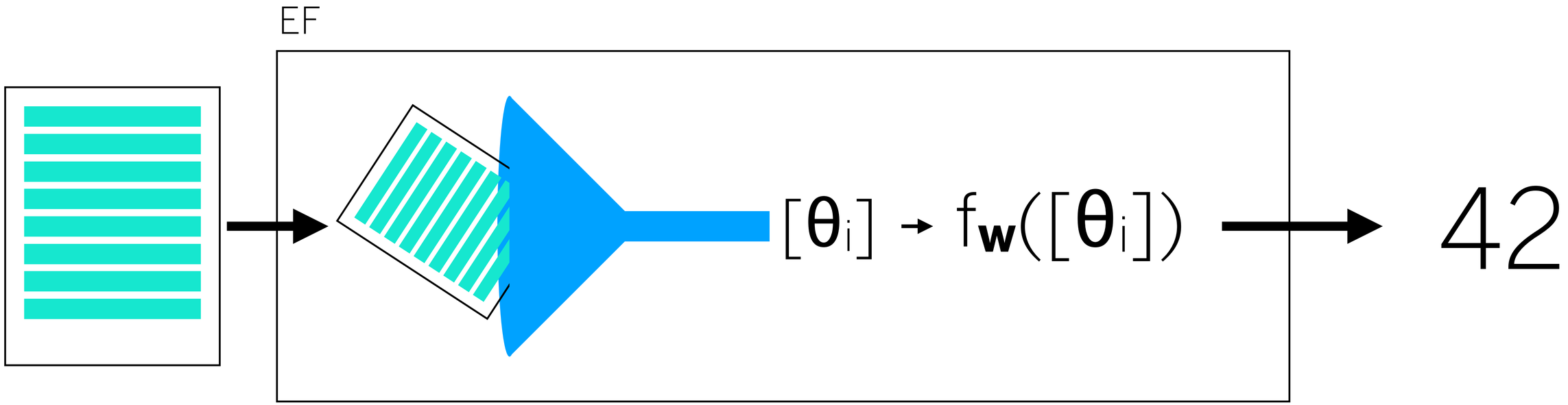
Mixer function

$$\mathbf{w} \cdot \boldsymbol{\theta}$$



capture importance and dependencies between features

Event-value Function (EF)



assign a value to the game's dynamics triggered

What about v(s) and q(s,a)?

State-value function **v(s)**

- ➡ assigns a value to a state of the game

State-action-value function **q(s,a)**

- ➡ assigns a value to the pair (state, action)

Drawbacks

- dimensionality
- the AI has to learn the important bits
- it is a *static* representation of the state
- state representation vary even within the same game

What is an event?

any modification to the game state

What is an event made of?

tick: when it happened;

who: who triggered it;

type: unique identifier of the type of event in the range [0, #types – 1];

duration: how long it lasted;

duration type: whether the event is instant, delayed or durative;

attributes: dictionary of attributes characterising the event;

signature: list of possible attribute keys;

trigger: what action triggered it.

What is an event made of?

tick: when it happened;

who: who triggered it;

type: unique identifier of the type of event in the range [0, #types – 1];

duration: how long it lasted;

duration type: whether the event is instant, delayed or durative;

attributes: dictionary of attributes characterising the event;

signature: list of possible attribute keys;

trigger: what action triggered it.

Splendor

The Game

2-Player Setup

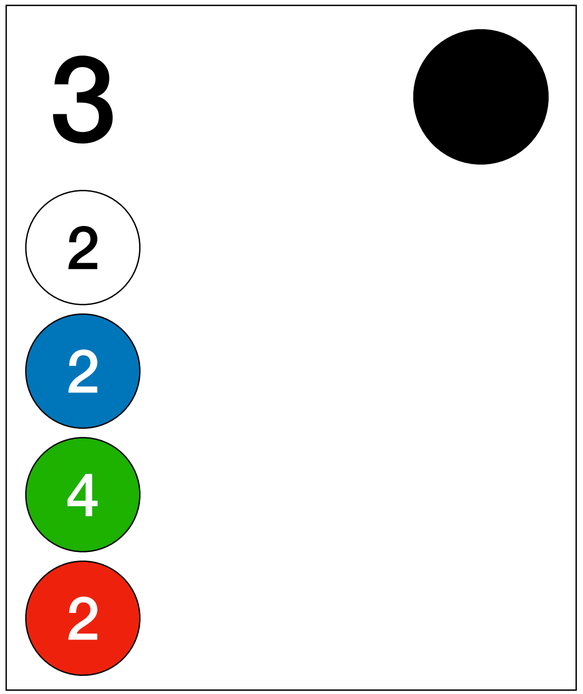
- 3 random nobles
- 3 decks
- 4 face-up cards per deck
- 5 common tokens per type
- 5 joker tokens

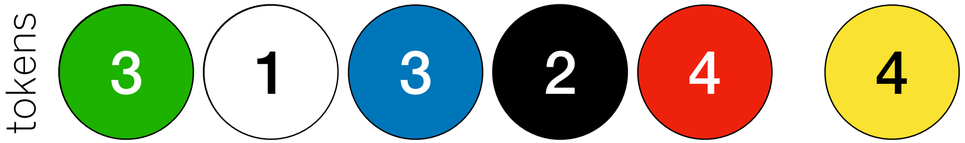
Event Logging

a simplified example

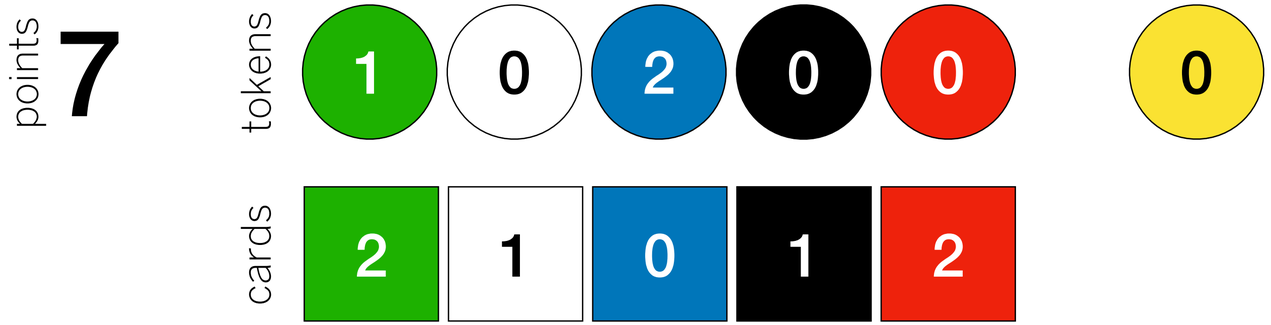
Current state

Table





Player 1

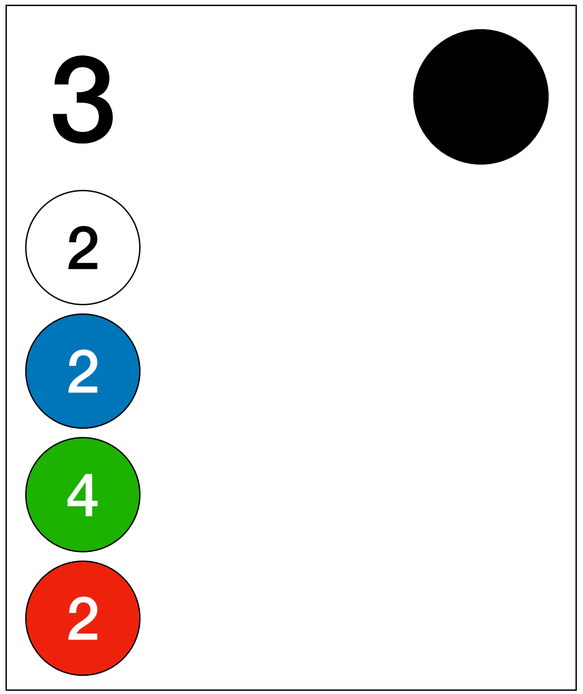


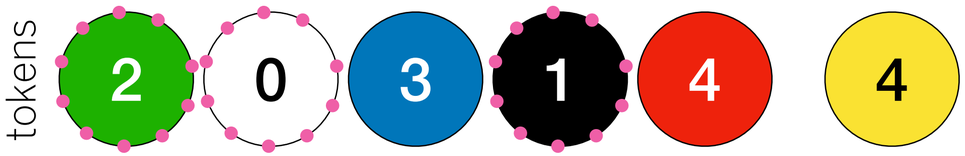
Log

{}

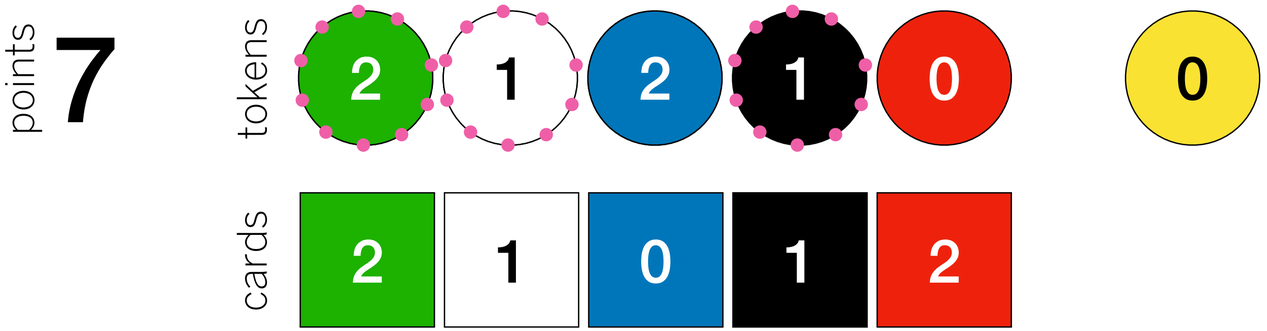
Action: pick 1 green, 1 white, 1 blue

Table





Player 1

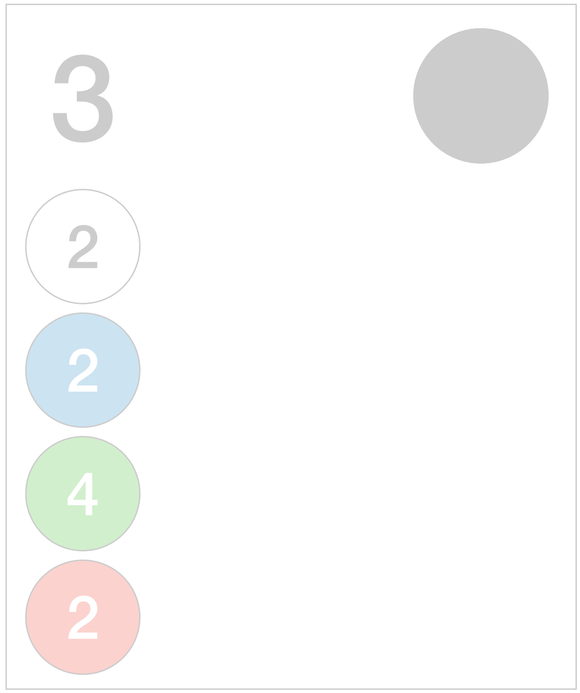


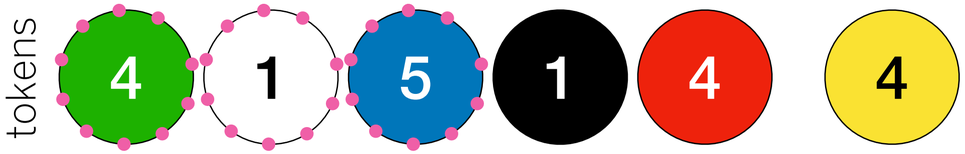
Log

{(2, P1), (8, P1),(2, P1),
(8, P1),(2, P1), (8, P1)}

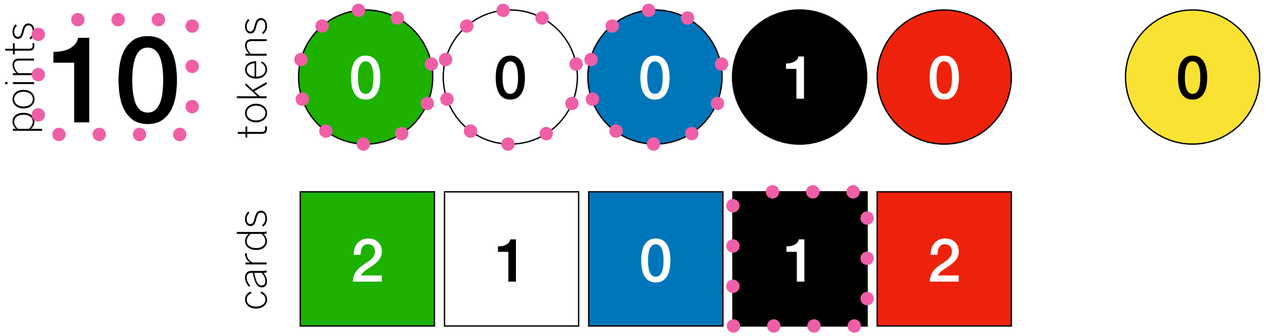
Action: buy card

Table





Player 1



Log

{(2, P1), (8, P1),(2, P1),
(8, P1),(2, P1), (8, P1),
(9,P1), (1,P1),(9,P1),
(1,P1), (9,P1), (1,P1),
(15,P1), (5,P1),(6,P1), (16,P1)}

Events in 𝔎

State element	Event	Who	Type^{id}	Type^{hc}
Noble	place, take, receive	P_i	7, 0, 14	-1, -1, 3
Table's token	increase, decrease	P_i	1, 2	-1, -1
Table's joker	increase, decrease	P_i	3, 4	-1, -1
Table's card	draw, place	P_i	5, 6	-1, -1
Player's token	increase, decrease	P_i	8, 9	0, -1
Player's joker	increase, decrease	P_i	10, 11	0, -1
Table's card	reserve, hidden	P_i	13, 12	2, 1
Player's points	from card	P_i	16	4
Player's points	from noble	E	17	4
Player's card	buy	P_i	15	-1

Events in 𝔎

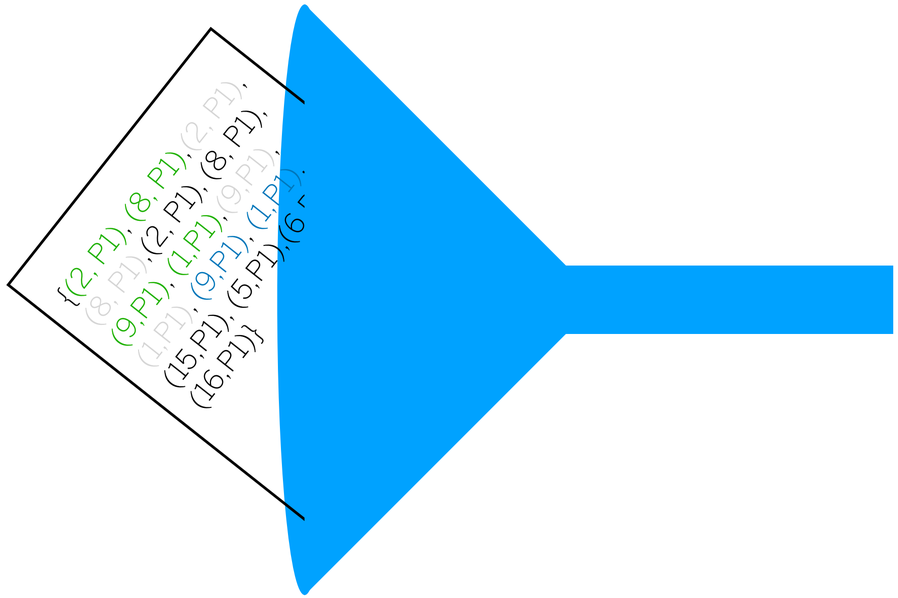
State element	Event	Who	Type^{id}	Type^{hc}
Noble	place, take, receive	P_i	7, 0, 14	-1, -1, 3
Table's token	increase, decrease	P_i	1, 2	-1, -1
Table's joker	increase, decrease	P_i	3, 4	-1, -1
Table's card	draw, place	P_i	5, 6	-1, -1
Player's token	increase, decrease	P_i	8, 9	0, -1
Player's joker	increase, decrease	P_i	10, 11	0, -1
Table's card	reserve, hidden	P_i	13, 12	2, 1
Player's points	from card	P_i	16	4
Player's points	from noble	E	17	4
Player's card	buy	P_i	15	-1

Events in 𝔎

State element	Event	Who	Type^{id}	Type^{hc}
Noble	place, take, receive	P_i	7, 0, 14	-1, -1, 3
Table's token	increase, decrease	P_i	1, 2	-1, -1
Table's joker	increase, decrease	P_i	3, 4	-1, -1
Table's card	draw, place	P_i	5, 6	-1, -1
Player's token	increase, decrease	P_i	8, 9	0, -1
Player's joker	increase, decrease	P_i	10, 11	0, -1
Table's card	reserve, hidden	P_i	13, 12	2, 1
Player's points	from card	P_i	16	4
Player's points	from noble	E	17	4
Player's card	buy	P_i	15	-1

Synthesis

Count action types!



Log (3 states, 2 actions)

{(2, P1), (8, P1),(2, P1),
(8, P1),(2, P1), (8, P1),
(9,P1), (1,P1),(9,P1),
(1,P1), (9,P1), (1,P1),
(15,P1), (5,P1),(6,P1),
(16,P1)}

[0,3,3,0,0,1,1,0,3,3,0,0,0,0,0,1,1,0]

18D (or **5D**)

State representation (1 state)

[10.0, 35.0, 26.0, 16.0, 4.0, 4.0, 4.0, 0.0, 0.0, 1.0, 0.0,
0.0, 4.0, 3.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 2.0, 2.0, 0.0, 1.0,
0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 2.0, 0.0, 2.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 3.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 2.0, 1.0, 1.0, 0.0, 0.0,
0.0, 0.0, 1.0, 2.0, 0.0, 0.0, 2.0, 3.0, 1.0, 0.0, 0.0, 0.0, 0.0,
1.0, 0.0, 3.0, 0.0, 2.0, 3.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 2.0,
3.0, 0.0, 3.0, 0.0, 2.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
5.0, 3.0, 0.0, 3.0, 0.0, 1.0, 0.0, 0.0, 0.0, 3.0, 0.0, 3.0, 3.0,
5.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 7.0, 3.0, 0.0,
3.0, 0.0, 0.0, 0.0, 0.0, 1.0, 3.0, 5.0, 3.0, 0.0, 3.0, 5.0, 0.0,
0.0, 1.0, 0.0, 0.0, 3.0, 0.0, 0.0, 0.0, 7.0, 0.0, 1.0, 2.0, 3.0,
1.0, 3.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0,
2.0, 3.0, 1.0, 3.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

174D

The AI

agents tested

Baseline Agents

Basic

RND: Random

OSLA: One-Step Look Ahead

Advanced

SRH*: tuned Seeded Rolling Horizon

BMRH*: tuned Branching Mutation Rolling Horizon

MCTS*: tuned Monte Carlo Tree Search

EF BMRH Agents

BMRH hyperparameter space

Symbol	Type	Description
l	integer	sequence length
n	integer	sequences evaluated
usb	boolean	if it uses shift buffer
mo	boolean	if it has to mutate once
ms	integer	mutation type
dcy	double	probability of exponential decay
μ	double	mean of the gaussian mutation point
σ	double	std dev of the gaussian mutation point

10D

+

EF hyperparameter space

Mixer	Features	Weights
$lin_{\mathbf{w}}^{hc}$	5	5
$poly_{\mathbf{w}}^{2,hc}$	5	15
$poly_{\mathbf{w}}^{3,hc}$	5	35
$lin_{\mathbf{w}}^{id}$	18	18
$poly_{\mathbf{w}}^{2,id}$	18	171
$poly_{\mathbf{w}}^{3,id}$	18	1140

+

[5, 15, 35, 18, 171, 1140]D

Nomenclature

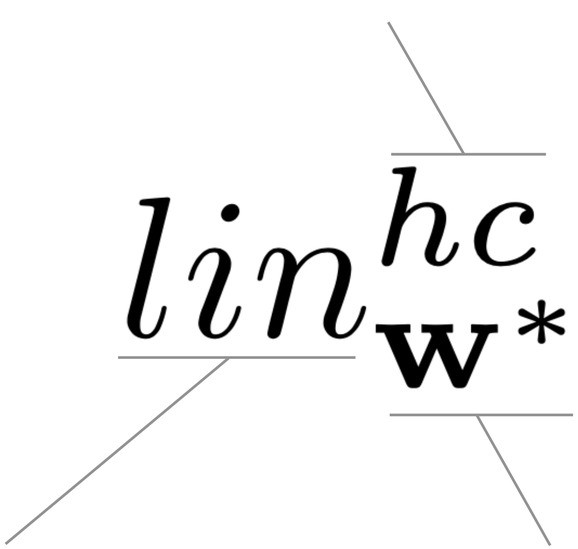
All agents are based on the Branching Mutation Rolling Horizon algorithm.

Only the event-value function changes.

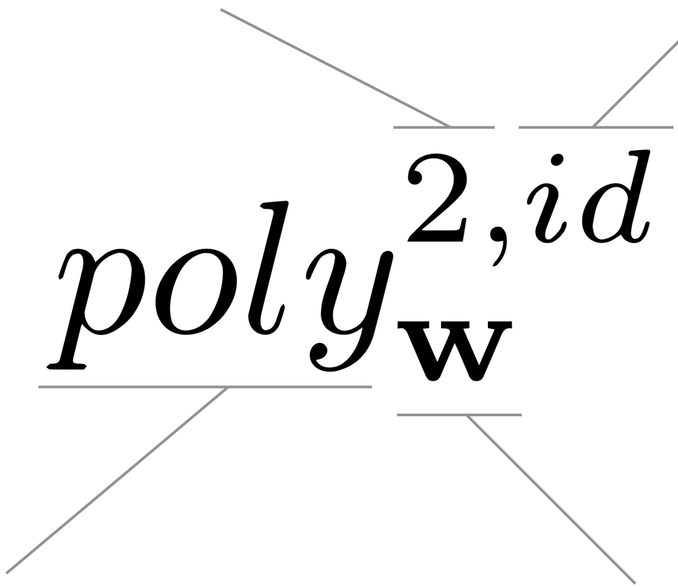
Hand crafted event ids

polynomial **grade**

Identity event ids



$$lin^{hc}_{\mathbf{w}*}$$



$$poly^{2,id}_{\mathbf{w}}$$

Linear mixer function

Tuned weights

Polynomial mixer function

Weights **space**

Experiments

ℜ to the test

The 3 experiments

① **Hyperparameter Tuning**, what is the best configuration for our agents?

② **Round Robin Tournament**, how do the best agents perform against each other?

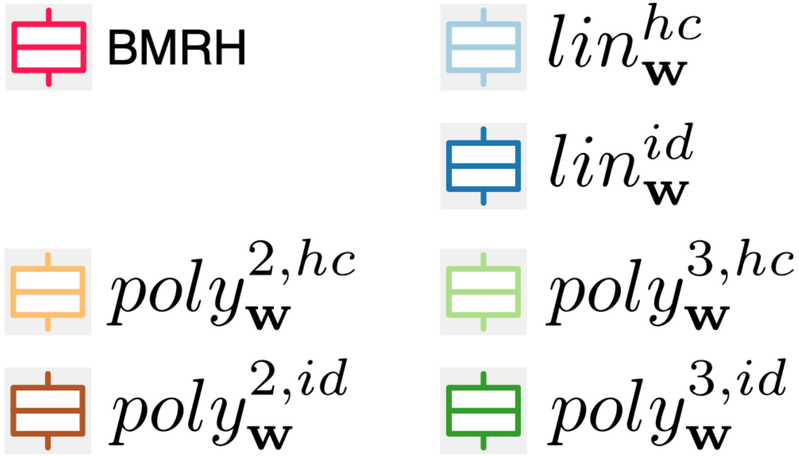
③ **Multi-opponent games**, how do they perform when against more than 1 opponent?

① Tuning

Setup

- play against BMRH*
- algorithm: NTBEA
- evaluation: single game
- validation: 1000 games

Agents



① Tuning

Setup

- play against BMRH*
- algorithm: NTBEA
- evaluation: single game
- validation: 1000 games

Agents

BMRH

$lin_{\mathbf{w}}^{hc}$

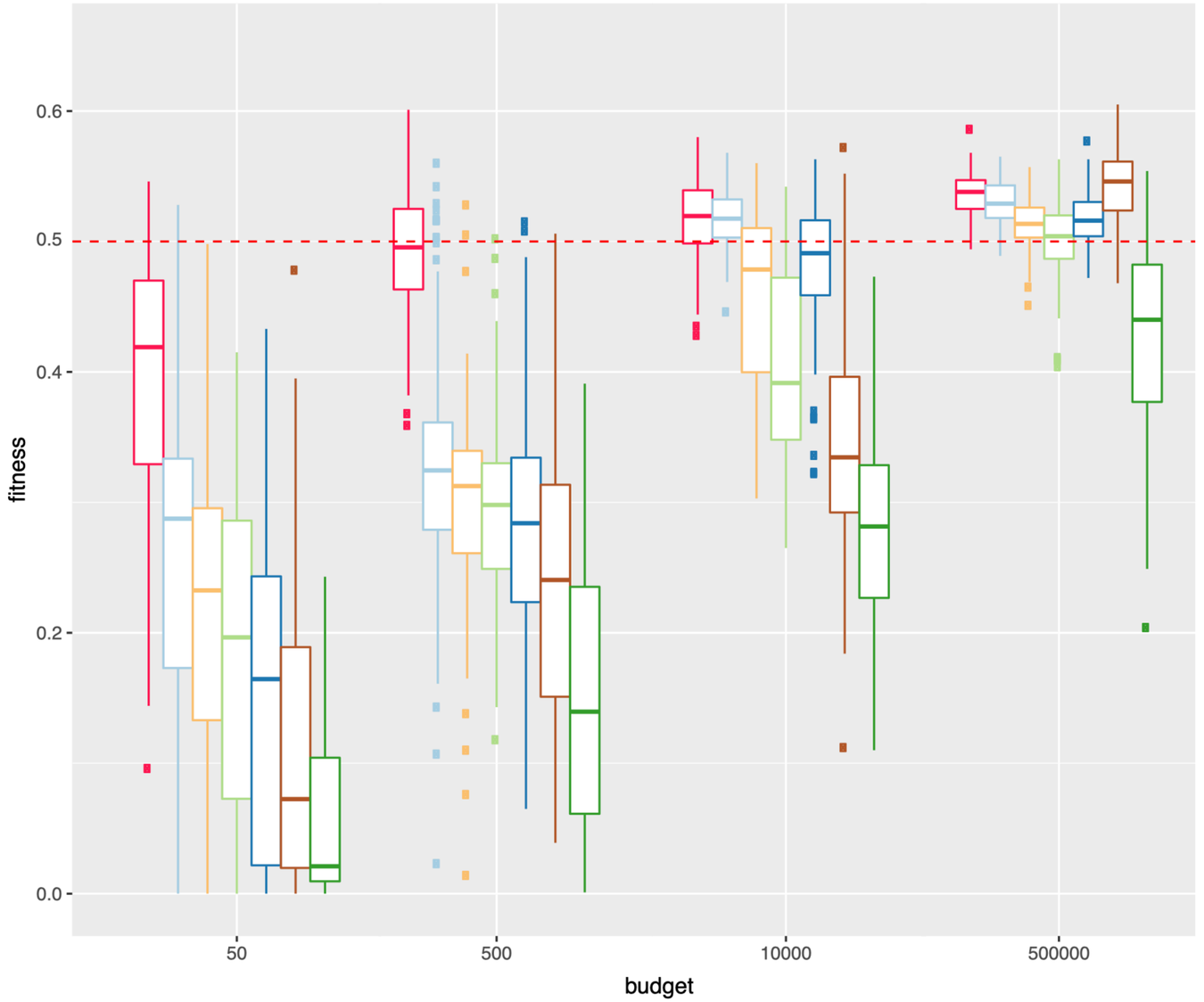
$lin_{\mathbf{w}}^{id}$

$poly_{\mathbf{w}}^{2,hc}$

$poly_{\mathbf{w}}^{3,hc}$

$poly_{\mathbf{w}}^{2,id}$

$poly_{\mathbf{w}}^{3,id}$



① Tuning

Setup

- play against BMRH*
- algorithm: NTBEA
- evaluation: single game
- validation: 1000 games

Agents

BMRH

$lin_{\mathbf{w}}^{hc}$

$lin_{\mathbf{w}}^{id}$

$poly_{\mathbf{w}}^{2,hc}$

$poly_{\mathbf{w}}^{3,hc}$

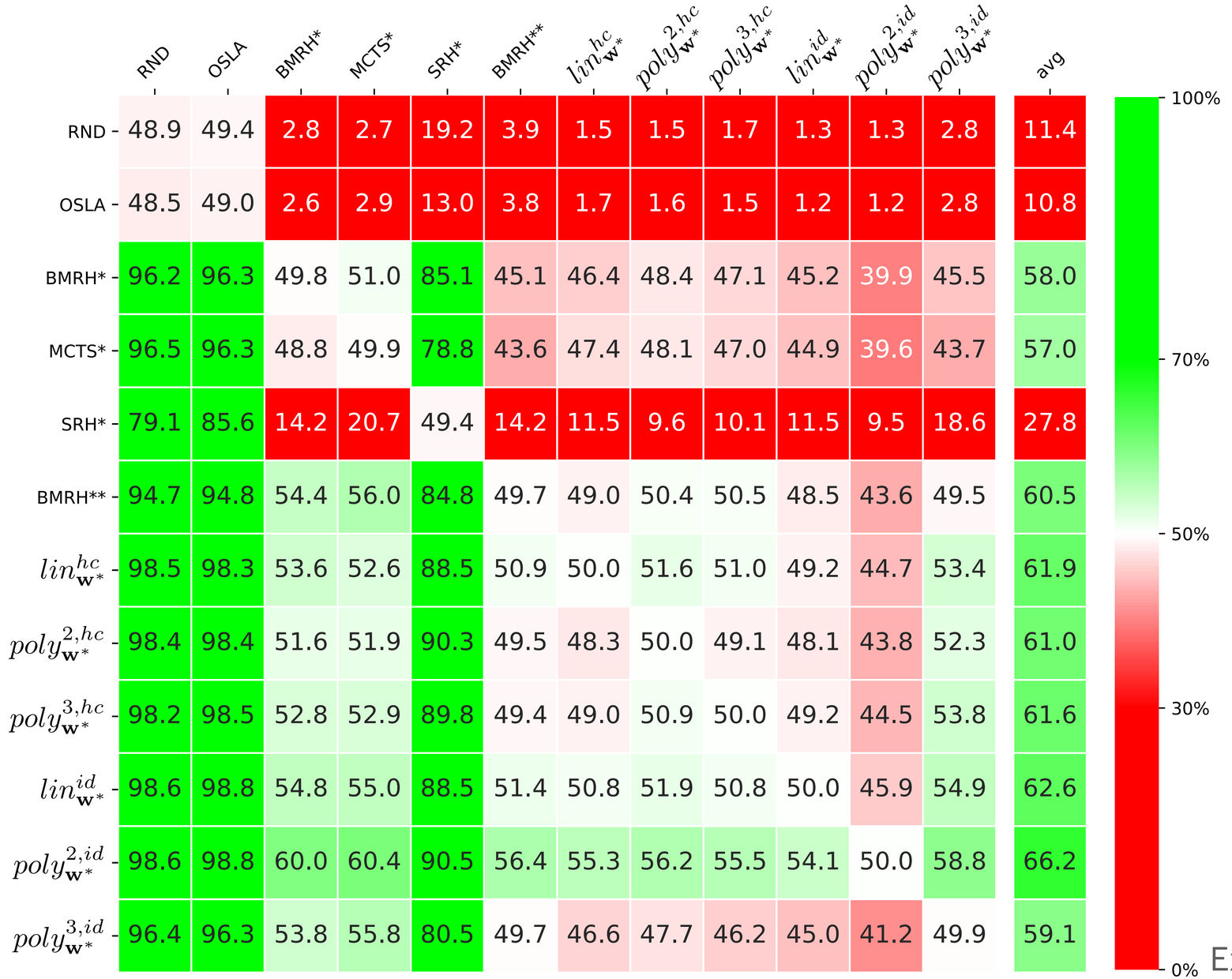
$poly_{\mathbf{w}}^{2,id}$

$poly_{\mathbf{w}}^{3,id}$

② Round Robin Tournament

Setup

- 10000 games each couple
- error $\pm 1\%$



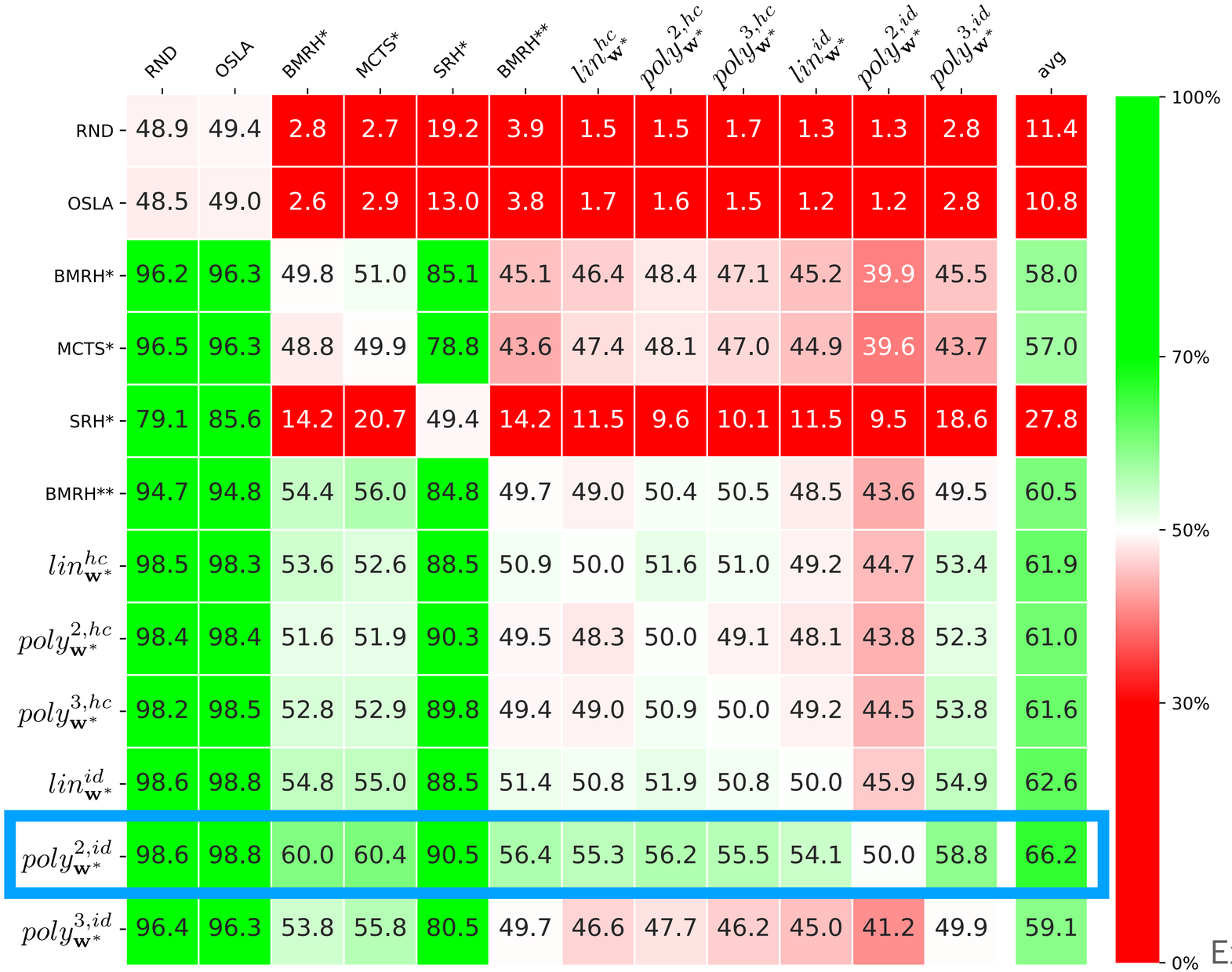
② Round Robin Tournament

Setup

- 10000 games each couple
- error $\pm 1\%$

$poly^{2,id}_{\mathbf{w}^*}$

the best across all agents



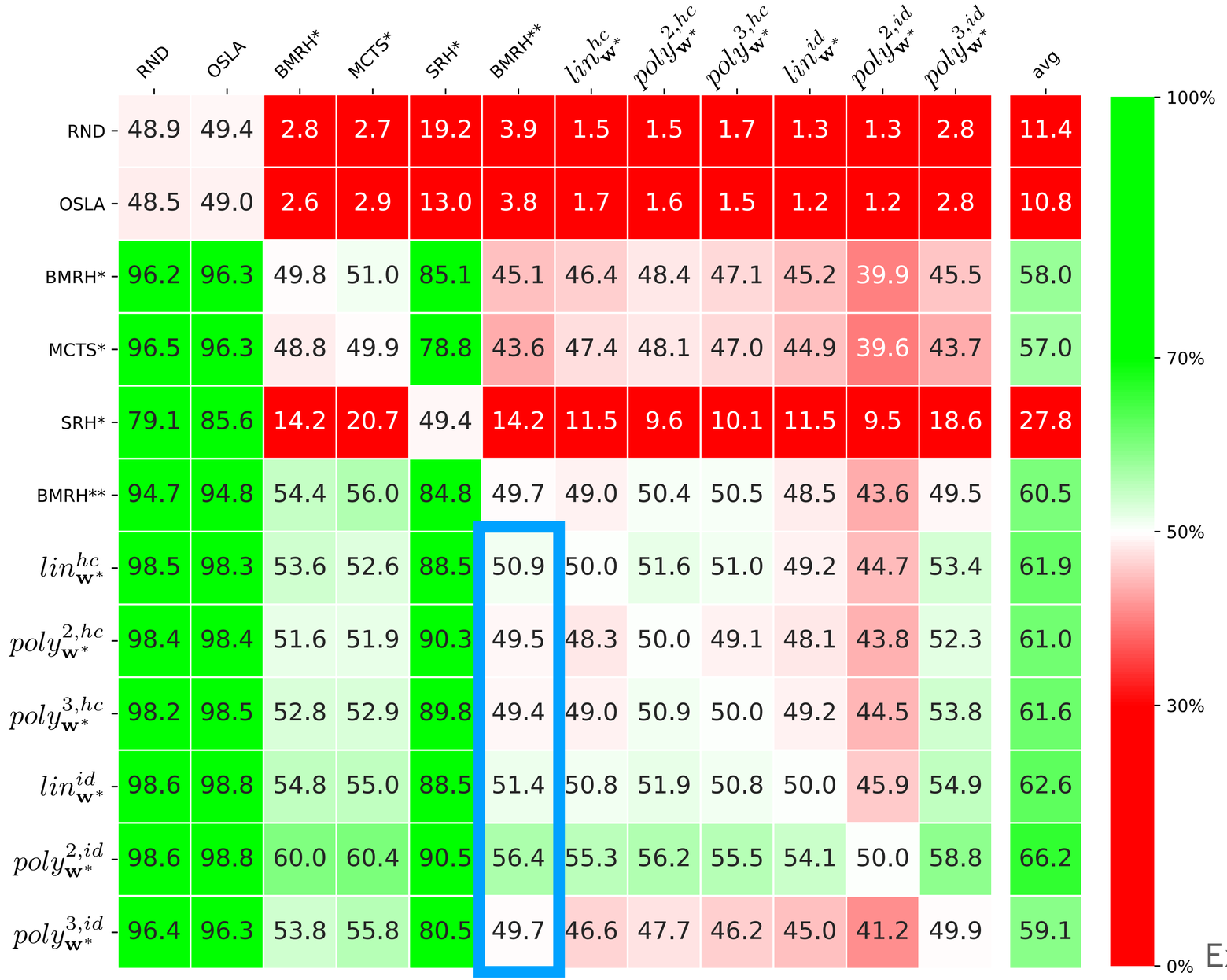
② Round Robin Tournament

Setup

- 10000 games each couple
- error $\pm 1\%$

as good as BMRH**

at worst



③ Multiple Opponents - Drop in win %

Behavioural Insights

let's look at the weights

Linear mixer with hand-crafted event ids

$lin^{hc}_{\mathbf{w}*}$

receive tokens, receive noble, receive points

$\mathbf{w}*$ (0.2, 0.2, -0.4, -0.6, 0.8)

buy card (3 pts) + get points = 0.8

3

buy card (0 pts) + attract noble (3 pts) + get points = - 0.6 + 0.8 = 0.2

3

Linear mixer with original event ids

$lin^{id}_{\mathbf{w}*}$

$\mathbf{w}*$ (-0.8, 0.2, -0.4, -1.0, 0.8, 0.2, -0.2, -0.2, 0.8, -0.2,
-1.0, -0.8, -0.8, 0.2, 1.0, 0.8, 1.0, 0.4)

give common token

give joker token

Summing up

what we have achieved

Have we succeeded?

We have improved the overall performance

Have we succeeded?

We have improved the overall performance

The agent now has deeper insight on its decision making

Have we succeeded?

We have improved the overall performance

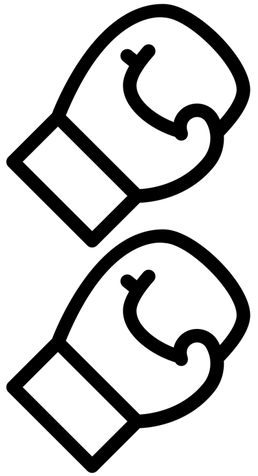
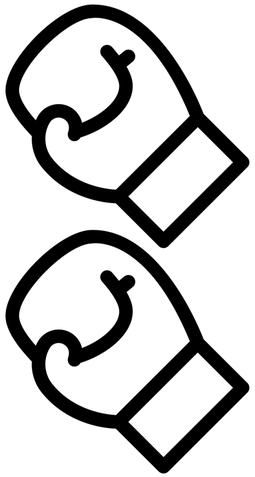
The agent now has deeper insight on its decision making

We learned something about the game!

Future work

what is around the corner

Compare to State-value Functions

EF   SF

More on behaviours

Deeper and more thorough analysis on behaviour

MAP-Elites vs RND search vs [imagine another algorithm here]

Design-assisting tool

Thanks

paper

@ivanbravi

i.bravi@qmul.ac.uk

Simplicity

clear discrete game state representation

simple actions

instant game events

Complexity

long term implications of early-game actions

limited game duration

hard to master

partial observability

multi-player

Parameters

#players	4	#decks *	3
#token types*	5	#extra nobles*	1
#coins per token	7	pick different #token types	3
#gold tokens	5	pick different #tokens	1
max #tokens	10	pick same #coins	2
end game prestige	15	pick same min #coins	4
#face-up cards	4	reserve max #cards	3

**requires Procedural Content Generation*

Dimensionality

14 dimensions

Classic 4 players Splendor

[4 , 5 , 7 , 5 , 3 , 4 , 1 , 10 , 15 , 3 , 1 , 2 , 4 , 3]

Action Space

Usually

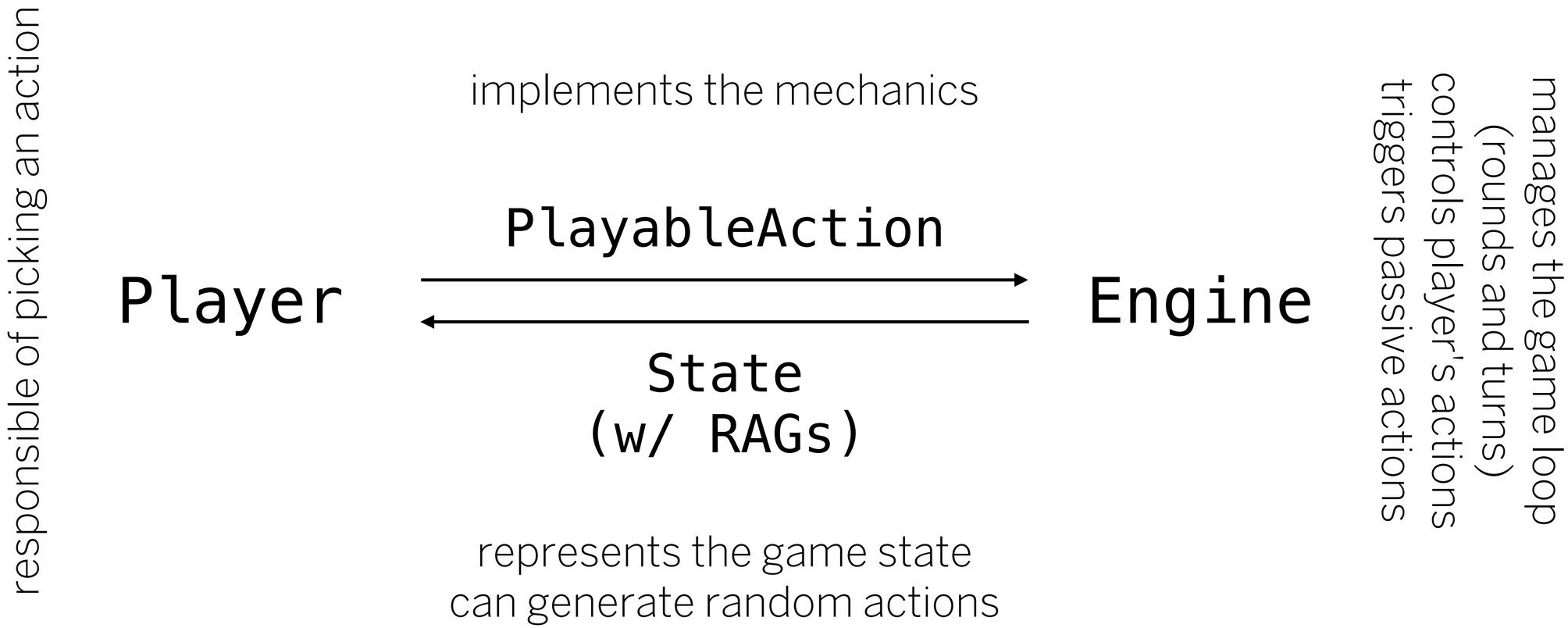
- count actions theoretical upper bound
- provide explicit action space

Action Space

Random Action Generator (RAG)

- each action type has a RAG that can generate random *legal* action
- it can be seeded to *control* the randomness
- samples the legal action space

Architecture



Performance

Running 10,000 games of 4 random players*

Stalemate frequency	1,410/10,000
Speed	~1.74M states/s
Average full game duration	0.44 ms

*6yo 2.3 GHz Intel Core i7