Imitation Learning over Heterogeneous Agents with Restraining Bolts

Students	ARCINIEGAS, Andrés (1874069)
	MAKINWA, Sayo (1858908)
	OREL, Egor (1836231)
	RUIZ, David (1922212)
Professor	DE GIACOMO, Giuseppe, PhD.

(De Giacomo et all, 2020)

TABLE OF CONTENTS

PROBLEM DESCRIPTION

A common problem in Reinforcement Learning (RL) is that the reward function is **hard to express.**

How to overcome?

Inverse Reinforcement Learning (IRL)

a set of execution traces (expert agent) → a reward function → learn the expert's behavior (learning agent)

Imitation Learning (IL)

PROBLEM DESCRIPTION

Typical IRL solutions

rely on a numerical representation of the reward function

- Problems related to the adopted optimization procedures;
- Must share the representation state (i.e. states and actions)

IL method

a logical (≠numerical) specification of the reward function

- No need of optimization procedures;
- Can be applied to heterogeneous agents, with the expert, the learner and the RB using different representations of the environment's actions and states, without specifying mappings among their representations.

Incorporated into a Restraining Bolt (RB)

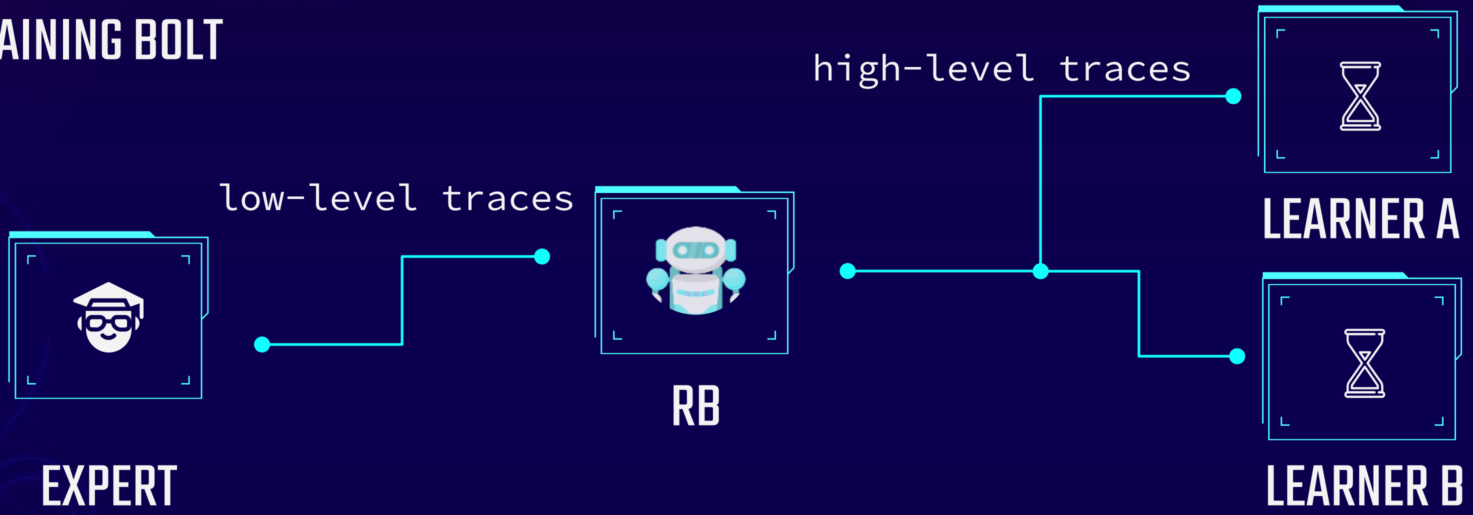
The RB can be attached to the learning agent to drive the learning process and ultimately make it imitate the expert.

RESTRAINING BOLT

RB is a device, with its own sensors, that can be attached to a Reinforcement Learning (RL) agent, to constrain its behavior and make it fulfill desired temporal high-level goals.

High-level goals are expressed as formulas of linear-time temporal logic over finite traces, LDL_f, over a set of fluents, generally different from the features used by the RL agent.

RESTRAINING BOLT



In other words, the low-level traces generated by the expert are **transformed into high-level traces from the RB sensors.** Once the high-level behavior is learned, this can be transferred to an agent with different capabilities.

This process can be seen as IRL at the RB representation level: instead of estimating the reward function, we **reconstruct the DFA** associated with the goal formula and then use it for learning.

RESTRAINING BOLT

A Restraining Bolt is a tuple

$$RB = \langle L,\ \{\phi_i, r_i\}_{i=1}^{m} \rangle$$

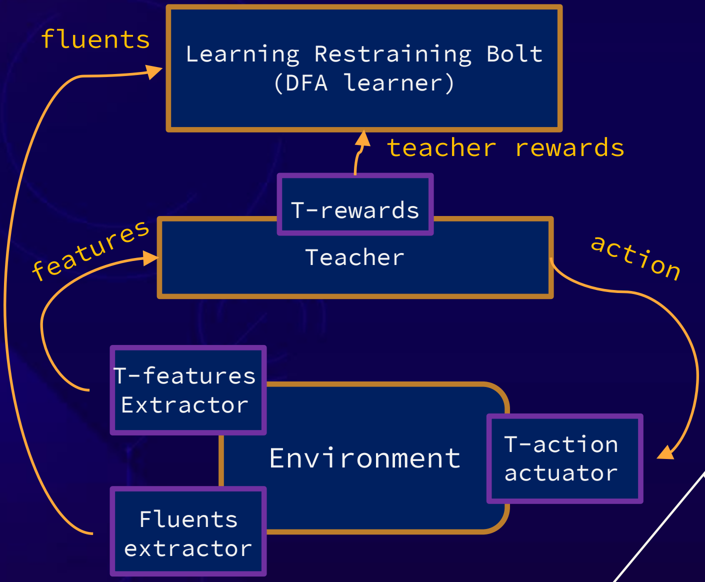
where:

- each ϕ_i is an LDL_f formula over a set of fluents L;
- each r_i is a reward value

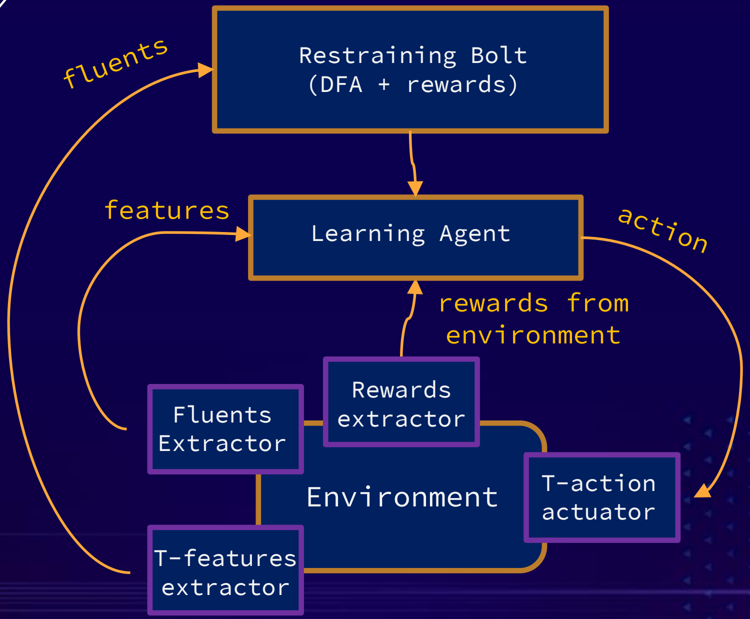
Formulas ϕ_i specify the behaviors that should be rewarded, each with its respective r_i.

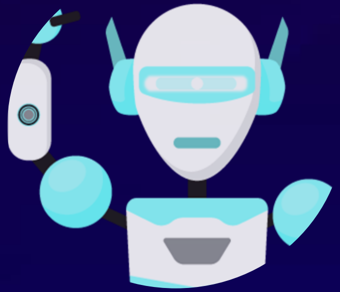
IMITATION LEARNING

Learn Restraining Bolt



Use Restraining Bolt





Deterministic Finite Automata

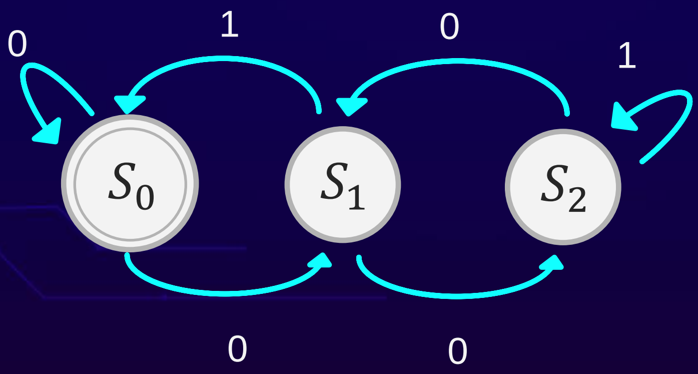
DFA

A DFA A is a tuple $A = (\Sigma, S, s^0, \rho, F)$, where

- Σ is a finite non-empty alphabet
- S is a finite non-empty set of states
- $s^0 \in S$ is the initial state
- $F \in S$ is the set of accepting states
- $\rho : S \times \Sigma \rightarrow S$ is a transition function

An automaton is essentially an edge-labelled directed graph:

States → Nodes
Transitions → Edges



RB DFA training

DFA = Deterministic Finite-state Automaton

Consider a setting where the expert agent executes its policy, producing desired and undesired traces at its own representation level.

negative (undesired) traces

positive (desired) traces

The resulting DFA represents (an approximation of) the expert's behavior.

Based on a well-known equivalence between LDL_f and DFAs, the DFA is incorporated into a RB attached to the RL agent, to make it learn a (possibly optimal) policy that imitates the expert's behavior.

We consider RBs of the form $\langle L, Q, r \rangle$ where Q is a DFA representing an LTL_f / LDL_f formula and r is a reward value associated with the accepting states of Q.

EXPERT AGENT

Consider an expert agent defined on a MDP:

$$M_e = \langle S_e, A_e, Tr_e, R_e \rangle$$

This agent knows how to act according to a policy, but **cannot describe its rewards**. The expert can correctly classify the traces as positive or negative, based on its own state representation.

Also, traces can be seen from RB perspective (RB sensors), so from each state fluents can be extracted to produce the corresponding representation in the RB space.

The expert **does not know anything about fluents**, in particular, it cannot interpret them, as belonging to a different representation space. In fact, the expert is not even aware of the RB.

LEARNING AGENT

Consider a learning agent defined on a MDP:

$$M_l=\langle S_l,A_l,Tr_l,R_l\rangle$$

Tr_l,R_l are unknown.
S_l,A_l and S_e,A_e respectively may be completely different, at least, without any explicit mapping between them.

The learning agent is also equipped with the RB that encodes the behavior of the expert agent in performing the given task.

The system M_l^{RB} can be used to learn an optimal policy driven by RB:

$$M_l^{RB}=\langle M_l,RB\rangle$$

In this way, the behavior of the learner agent imitates that of the expert, when considering the evolution at the RB level.

SOLUTION METHOD

The core problem: extracting the DFA (or the formula) from the set T of (positive and negative) traces.

Notice that the target DFA is unknown, even to the expert. As a result, the best we can hope for is to come up with a good approximation. For this reason, we search for a DFA that accepts all positive and no negative traces, according to T.

GENERALIZATION OF TRACES

The best generalization of DFA over traces T:
- accepts more traces than exactly the positive ones;
- possibly reject more than the negative ones;
- + some bias.

One reasonable approach is to check smaller DFAs (in terms of number of states) first:
- smaller DFAs are less selective;
- tends to accept more traces than those with a large number of states.

EXTRACTING DFA

Several approaches exist for extracting a DFA from a set of labelled traces. In our case, any is a reasonable candidate. For simplicity, we have selected L*(Angluin 1987).

Why:

- Firstly, the algorithm returns a DFA (not a formula) that can be used as-is when executing the RB.
- Secondly, the algorithm produces increasingly larger DFAs, thus satisfying the generalization requirement discussed above – though it is not guaranteed to return the minimal DFA.

L* ALGORITHM

LEARNER

membership query ("is this a positive trace?") and equivalence query ("is this the target formula/automaton?")

EXPERT

answers: "yes", "no", "yes/no"

LEARNER

uses membership queries to produce a candidate DFA

LEARNER TO EXPERT

"Is this DFA a target one?"

"no"

Create a counterexample

"yes"

DFA is ready

The algorithm is shown to terminate and find the target DFA in polynomial time wrt both the size of the minimal DFA equivalent to the target DFA and the maximum length of any returned counterexample

L*

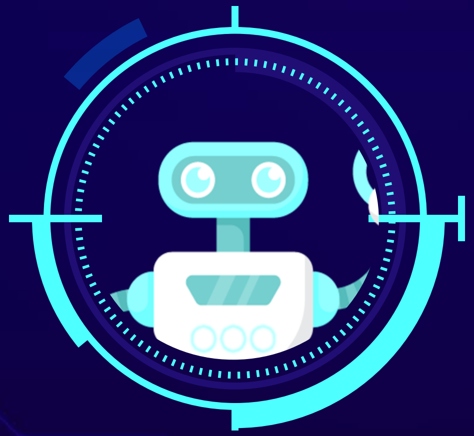
ALGORITHM

In our case, the expert executes the policy offline, so it can't check and answer, can't know whether the resulting DFA is target.

Also, an expert can't be an oracle and answer to all questions of the L*. But having a set of traces T executed by the expert learner the "oracle" can be simulated such that L* creates an approximation of the suitable DFA.

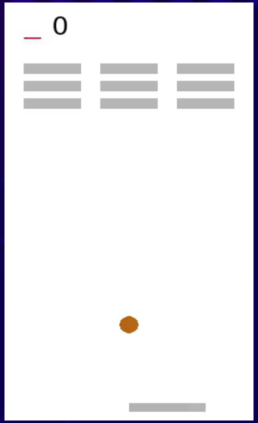
How is it done:
Learner poses a membership query;
Simulated "oracle":
"no" if T_i is negative;
"yes" if T_i is positive;
Learner: equivalence check is positive if the candidate DFA accepts all positive and no negative traces from T.

Case studies

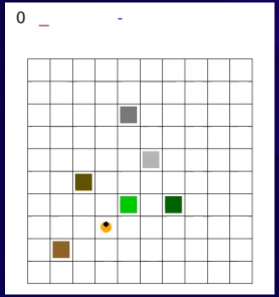


CASE STUDIES

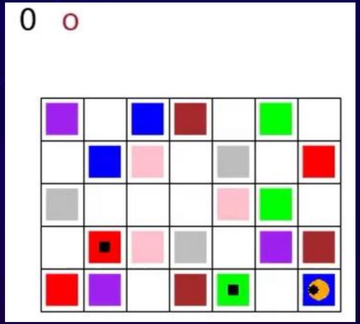
BREAKOUT



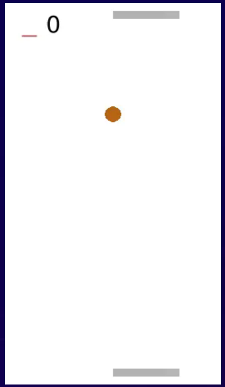
MINECRAFT



SAPIENTINO



PONG

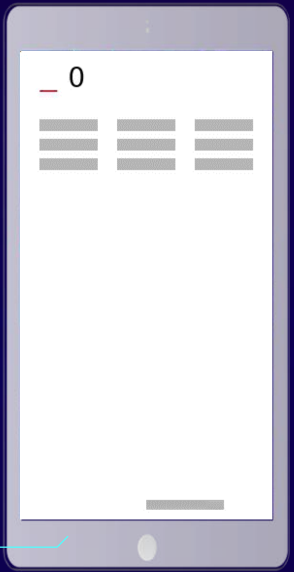


Case study: Breakout

Target task:

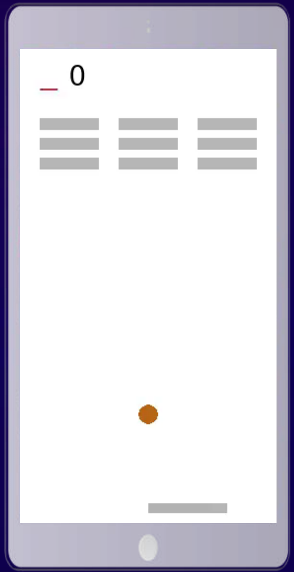
All bricks must be removed, completing the columns from left to right.

EXPERT



Uses **FIRE.**
Simple (easier) environment

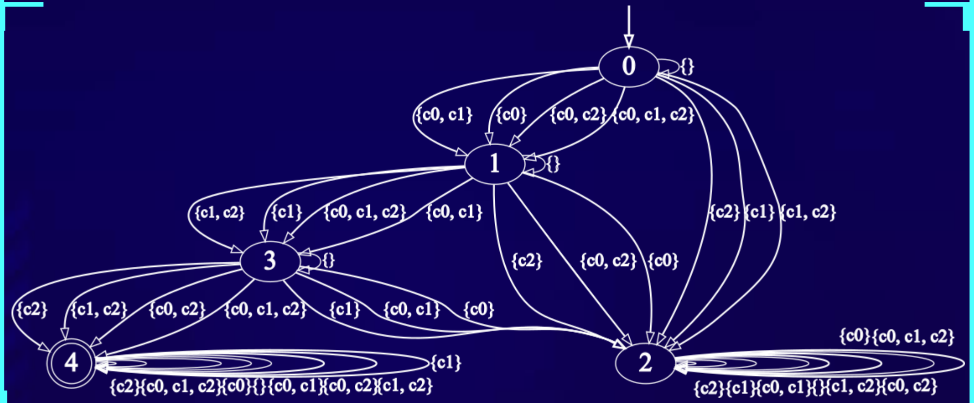
LEARNER



Uses **BALL.**
More complex environment

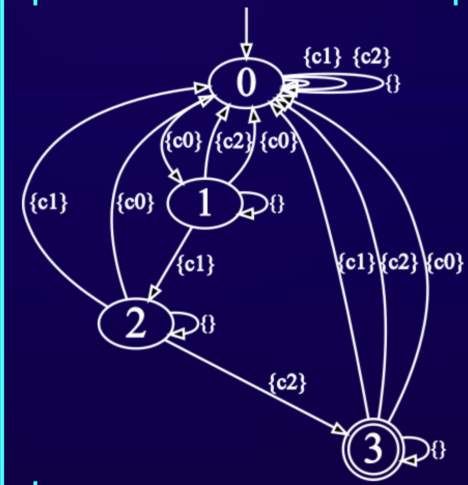
Breakout

TRUE AUTOMATON



From LDLf Formula

EXTRACTED AUTOMATON



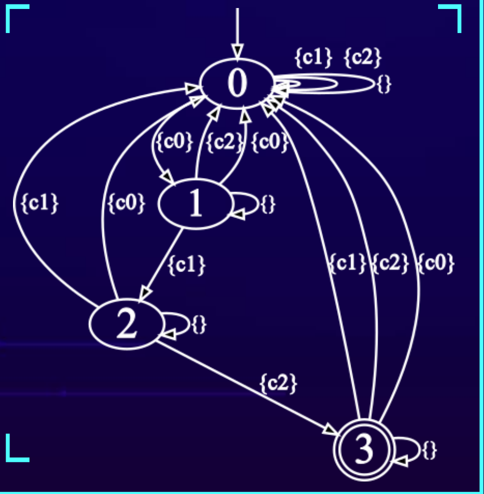
(BY THE EXPERT)

TRACES (samples)

POSITIVE TRACES	NEGATIVE TRACES
c0;c1;c2	c1;c0;c2
c0;c1;c2	c1;c2
c0;c1;c2	c1;c2
c0;c1;c2	c0;c2;c1
c0;c1;c2	c2;c1
c0;c1;c2	c1;c2;c0
c0;c1;c2	c2;c1
c0;c1;c2	c1;c2;c0
c0;c1;c2	c2;c1;c0
c0;c1;c2	c2;c1
c0;c1;c2	c1;c0
c0;c1;c2	c1;c0
c0;c1;c2	c0;c2;c1
c0;c1;c2	c2;c1;c0
c0;c1;c2	c2;c1;c0
c0;c1;c2	c2;c0

DFA EXTRACTION

States	S	0, 1, 2, 3
Alphabet	Σ	{}, c1, c2, c3
Initial State	s^0	0
Transition function	ρ	'0': {{c2}: '0', {c0}: '1', {c1}: '0', {}: '0'}, '1': {{c2}: '0', {c0}: '0', {c1}: '2', {}: '1'}, '2': {{c2}: '3', {c0}: '0', {c1}: '0', {}: '2'}, '3': {{c2}: '0', {c0}: '0', {c1}: '0', {}: '3'},
Accepting states	F	{3}



input to

Learner

Breakout

No direct relation from the Original Goal definition and the final learner

True DFA - Goal

Generated from the LDL formula, and describes the goal of the task.

Expert training

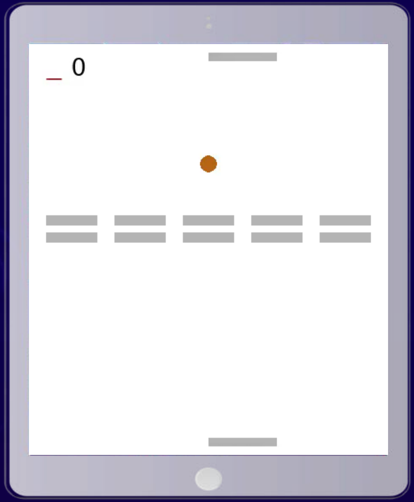
Generates DFA from traces

Learner training

with RB from generated DFA

Expert and Learner are trained with value-based RL techniques such **as Q-Learning and SARSA**

Case study: Breakout-**Pong**

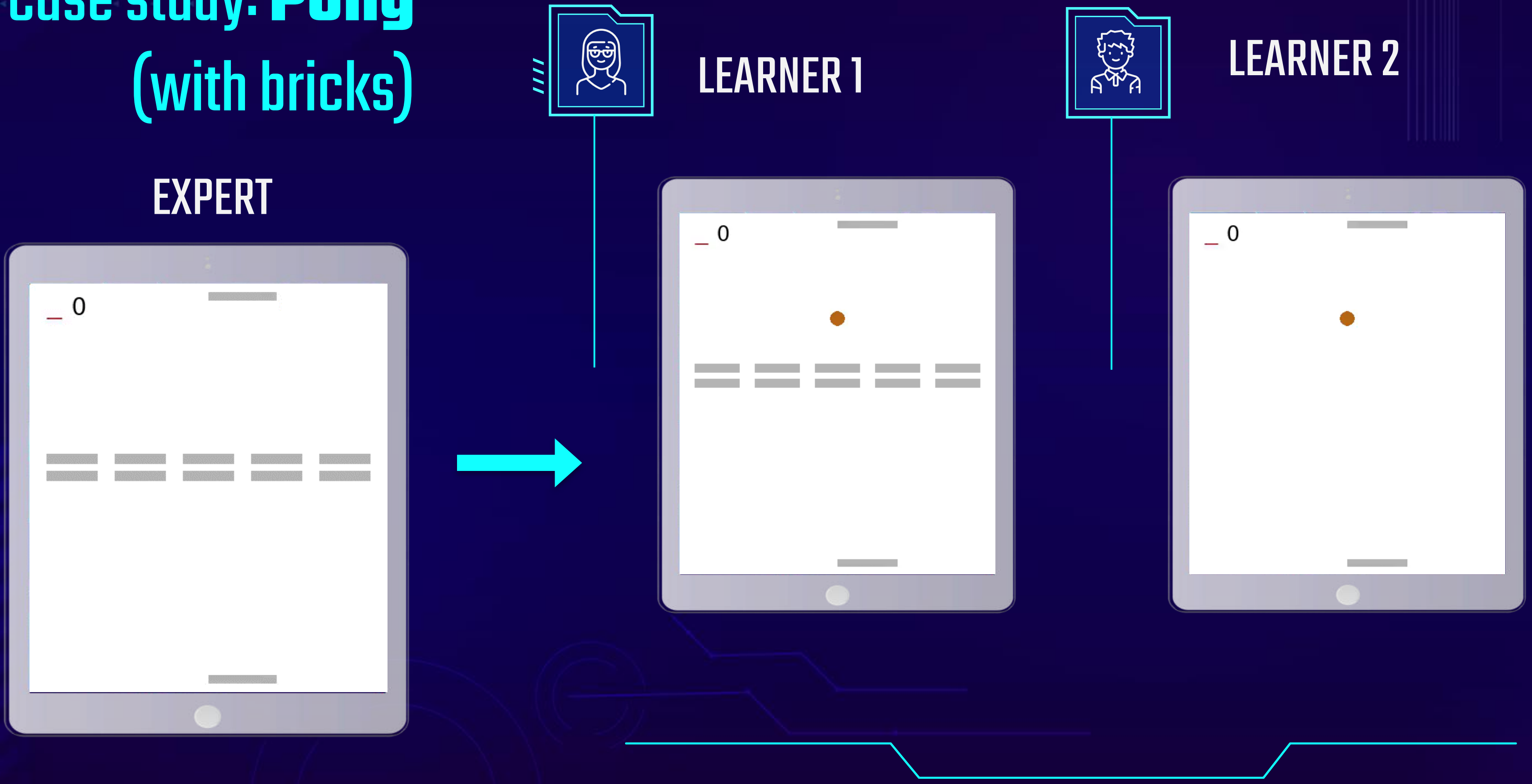


Goal remains the same: eliminate all the bricks.

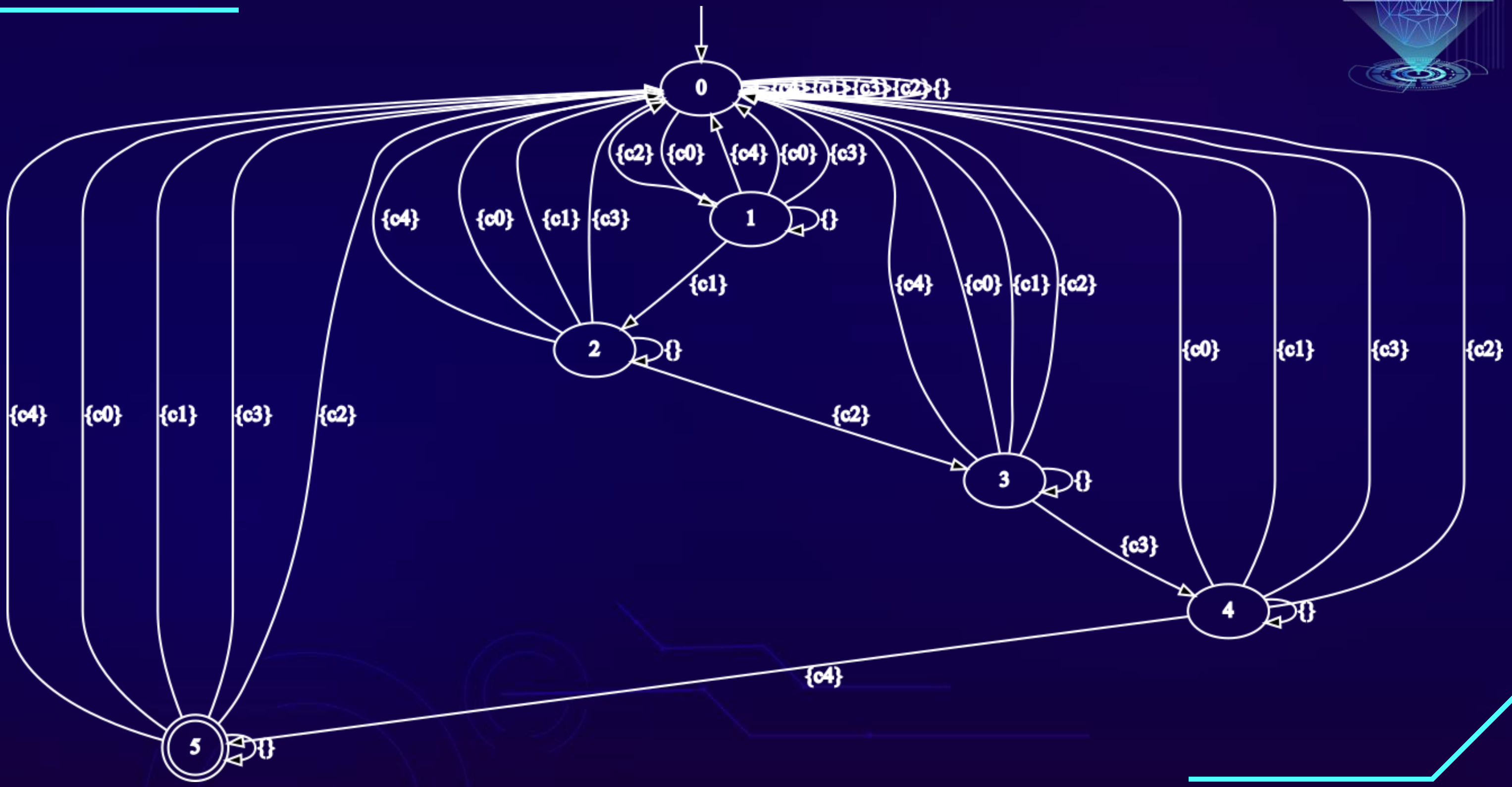
Addition of a paddle on the top that must play as well. Moves just as the bottom one.

Base for competitive Pong game.

Case study: **Pong** (with bricks)

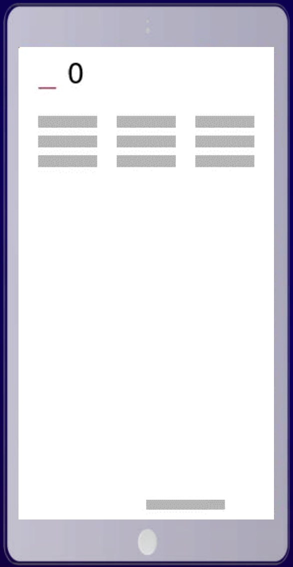


Generated DFA in 5-columns Breakout-Pong



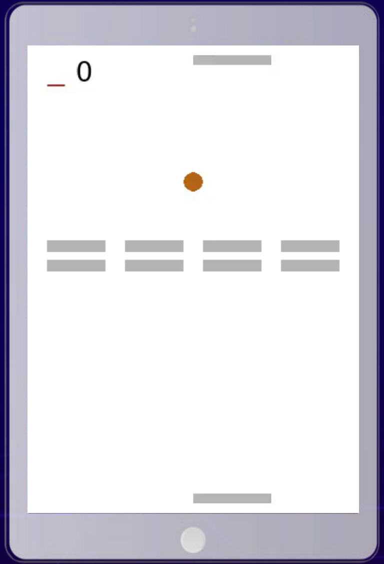
Additional variants:

Side change

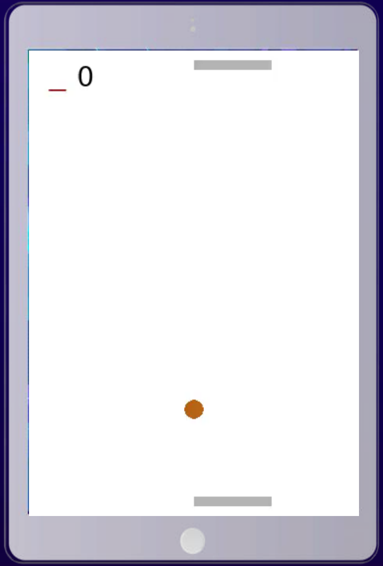


(in goal)

Breakout-pong 4 columns:



Pong 4 columns:



Conclusions

- It is an approach based on the use of Restraining Bolts to perform Imitation Learning, with ***heterogeneous agents.***
- It makes use of **Inverse Reinforcement Learning**, to represent as a **DFA the behavior** from the expert.
- DFA constitutes a **logical representation of the reward function**, avoiding problems with numerical representation.
- In all evaluated cases, the **task is successfully transferred** from expert to learner, despite the **differences in the state-action** representation space.

REFERENCES

Imitation Learning over Heterogeneous Agents with Restraining Bolts
Giuseppe De Giacomo, Marco Favorito, Luca Iocchi, Fabio Patrizi.
ICAPS 2020.

Foundations for Restraining Bolst: Reinforcement Learning with LTLf/LDLf restraining specifications
Giuseppe De Giacomo, Luca Iocchi, Marco Favorito, Fabio Patrizi.
ICAPS 2019

DFA, NFA, AFW on finite words
Giuseppe De Giacomo
Topic revisit