Foundations for Restraining Bolts: Reinforcement Learning with LTL_f/LDL_f restraining specifications

Reasoning Agents

SAPIENZA
UNIVERSITÀ DI ROMA

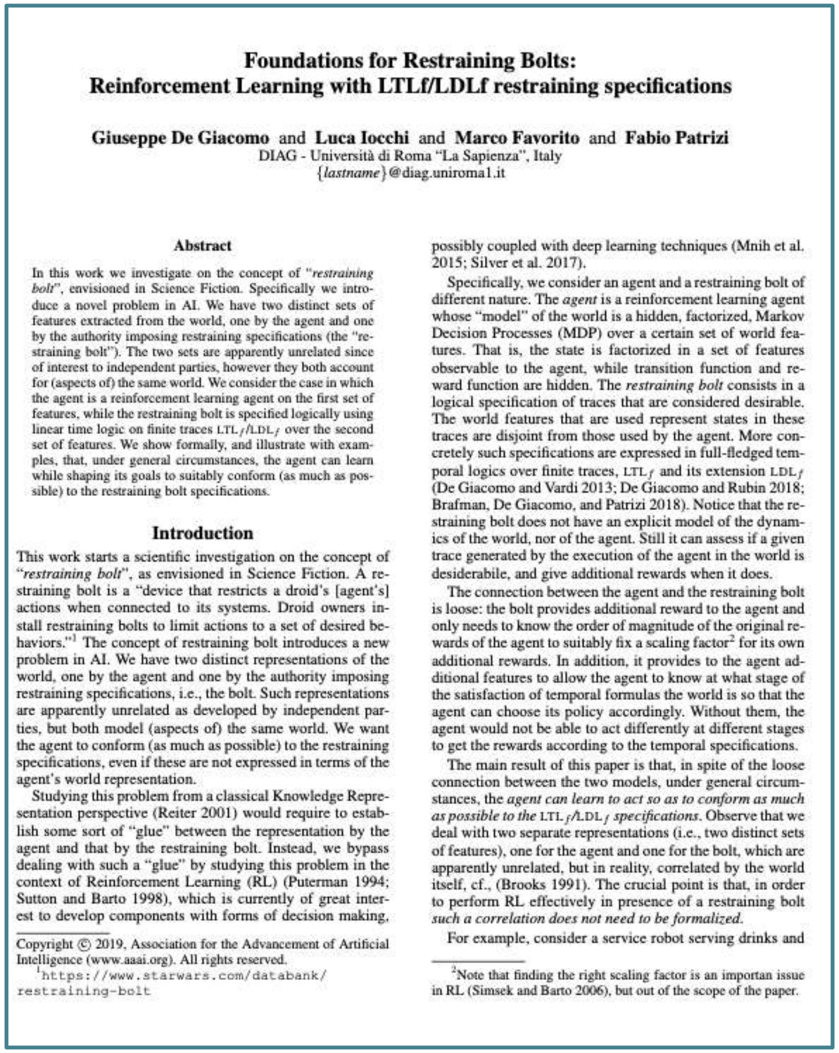
Sara Tozzo
1483104

Giorgia Piernoli
1648511

Flavio Lorenzi
1662963

Nicolò Mantovani
1650269

Papers



Foundations for Restraining Bolts: Reinforcement Learning with LTLf/LDLf restraining specifications

Giuseppe De Giacomo and **Luca Iocchi** and **Marco Favorito** and **Fabio Patrizi**
DIAG - Università di Roma "La Sapienza", Italy
{*lastname*}@diag.uniroma1.it

Abstract

In this work we investigate on the concept of "*restraining bolt*", envisioned in Science Fiction. Specifically we introduce a novel problem in AI. We have two distinct sets of features extracted from the world, one by the agent and one by the authority imposing restraining specifications (the "restraining bolt"). The two sets are apparently unrelated since of interest to independent parties, however they both account for (aspects of) the same world. We consider the case in which the agent is a reinforcement learning agent on the first set of features, while the restraining bolt is specified logically using linear time logic on finite traces $\text{LTL}_f/\text{LDL}_f$ over the second set of features. We show formally, and illustrate with examples, that, under general circumstances, the agent can learn while shaping its goals to suitably conform (as much as possible) to the restraining bolt specifications.

Introduction

This work starts a scientific investigation on the concept of "*restraining bolt*", as envisioned in Science Fiction. A restraining bolt is a "device that restricts a droid's [agent's] actions when connected to its systems. Droid owners install restraining bolts to limit actions to a set of desired behaviors."[1] The concept of restraining bolt introduces a new problem in AI. We have two distinct representations of the world, one by the agent and one by the authority imposing restraining specifications, i.e., the bolt. Such representations are apparently unrelated as developed by independent parties, but both model (aspects of) the same world. We want the agent to conform (as much as possible) to the restraining specifications, even if these are not expressed in terms of the agent's world representation.

Studying this problem from a classical Knowledge Representation perspective (Reiter 2001) would require to establish some sort of "glue" between the representation by the agent and that by the restraining bolt. Instead, we bypass dealing with such a "glue" by studying this problem in the context of Reinforcement Learning (RL) (Puterman 1994; Sutton and Barto 1998), which is currently of great interest to develop components with forms of decision making, possibly coupled with deep learning techniques (Mnih et al. 2015; Silver et al. 2017).

Specifically, we consider an agent and a restraining bolt of different nature. The *agent* is a reinforcement learning agent whose "model" of the world is a hidden, factorized, Markov Decision Processes (MDP) over a certain set of world features. That is, the state is factorized in a set of features observable to the agent, while transition function and reward function are hidden. The *restraining bolt* consists in a logical specification of traces that are considered desirable. The world features that are used represent states in these traces are disjoint from those used by the agent. More concretely such specifications are expressed in full-fledged temporal logics over finite traces, LTL_f and its extension LDL_f (De Giacomo and Vardi 2013; De Giacomo and Rubin 2018; Brafman, De Giacomo, and Patrizi 2018). Notice that the restraining bolt does not have an explicit model of the dynamics of the world, nor of the agent. Still it can assess if a given trace generated by the execution of the agent in the world is desiderabile, and give additional rewards when it does.

The connection between the agent and the restraining bolt is loose: the bolt provides additional reward to the agent and only needs to know the order of magnitude of the original rewards of the agent to suitably fix a scaling factor[2] for its own additional rewards. In addition, it provides to the agent additional features to allow the agent to know at what stage of the satisfaction of temporal formulas the world is so that the agent can choose its policy accordingly. Without them, the agent would not be able to act differently at different stages to get the rewards according to the temporal specifications.

The main result of this paper is that, in spite of the loose connection between the two models, under general circumstances, the *agent can learn to act so as to conform as much as possible to the $\text{LTL}_f/\text{LDL}_f$ specifications*. Observe that we deal with two separate representations (i.e., two distinct sets of features), one for the agent and one for the bolt, which are apparently unrelated, but in reality, correlated by the world itself, cf., (Brooks 1991). The crucial point is that, in order to perform RL effectively in presence of a restraining bolt *such a correlation does not need to be formalized*.

For example, consider a service robot serving drinks and

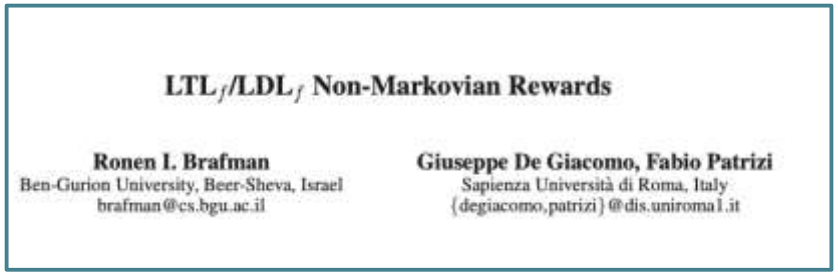
[1]https://www.starwars.com/databank/restraining-bolt

[2]Note that finding the right scaling factor is an important issue in RL (Simsek and Barto 2006), but out of the scope of the paper.

Non-Markovian Rewards Expressed in LTL: Guiding Search via Reward Shaping

Alberto Camacho,[2] Oscar Chen,[*3] Scott Sanner,[1] Sheila A. McIlraith[2]
[2]Department of Computer Science, University of Toronto
[1]Department of Mechanical & Industrial Engineering, University of Toronto
[3]Department of Engineering, University of Cambridge
[2]{acamacho,sheila}@cs.toronto.edu, [3]ozbc2@cam.ac.uk, [1]ssanner@mie.utoronto.ca

+



$\text{LTL}_f/\text{LDL}_f$ Non-Markovian Rewards

Ronen I. Brafman
Ben-Gurion University, Beer-Sheva, Israel
brafman@cs.bgu.ac.il

Giuseppe De Giacomo, Fabio Patrizi
Sapienza Università di Roma, Italy
{degiacomo,patrizi}@dis.uniroma1.it

Experimental part with some practical examples

🟩 🟨 ♞ ♝

Introduction

Restraining bolt: “a device that restricts a droid’s actions when connected to its systems. Droid owners install restraining bolts to limit actions to a set of desired behaviors.”

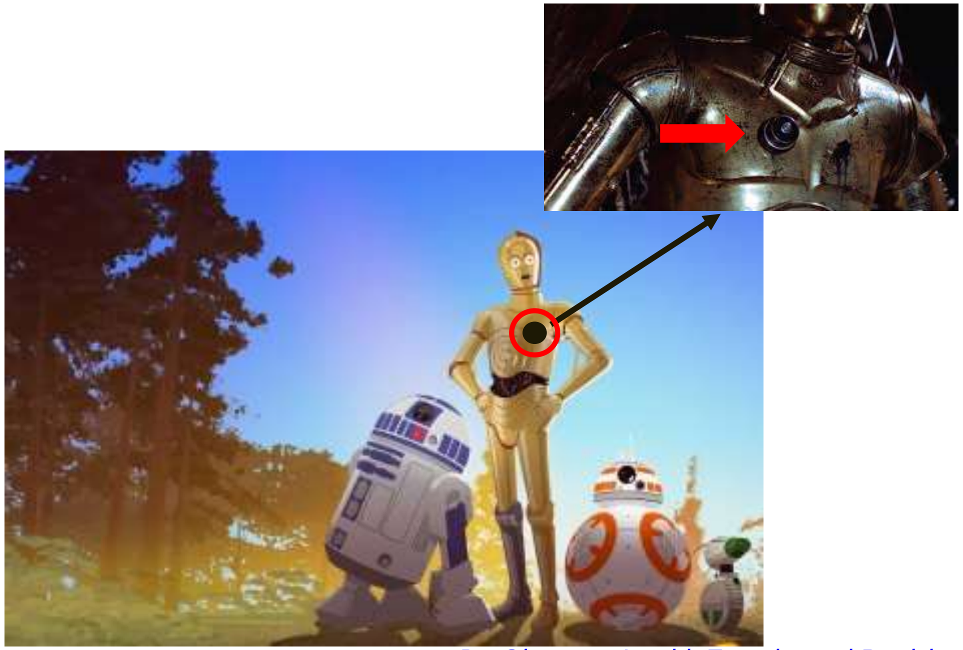
De Giacomo, Iocchi, Favorito and Patrizi, Foundations for Restraining Bolts: Reinforcement Learning with LTLf/LDLf restraining specifications. 2019

Problem for Artificial Intelligence

Two different representation of the world, apparently unrelated:

Agent: a *reinforcement learning agent* modeled by a *Markov Decision Process* (MDP)

Restraining Bolt (RB): logical specification of traces that are expressed in temporal logics over finite traces LTL_f /LDL_f



De Giacomo, Iocchi, Favorito and Patrizi, Foundations for Restraining Bolts: Reinforcement Learning with LTLf/LDLf restraining specifications. 2019

Goal

The bolt provides an additional reward and features to the agent.

↓

The agent is able to act at different stages to get the rewards according to the temporal specifications.

De Giacomo, Iocchi, Favorito and Patrizi, Foundations for Restraining Bolts: Reinforcement Learning with LTLf/LDLf restraining specifications. 2019

MDP: Markov Decision Process

It is a tuple $\mathrm{M} =< \mathrm{S}, \mathrm{A}, \mathrm{Tr}, \mathrm{R} >$ where:

- S is a set of states
- A is a set of actions
- $T_r : S \times A \rightarrow Prob(S)$ is the transition function
- $R : S \times A \times S \rightarrow \mathbb{R}$ is the reward function

A solution to an MDP is a policy ρ, assigns an action to each state, possibly conditioned on past states and actions.

De Giacomo, Iocchi, Favorito and Patrizi, Foundations for Restraining Bolts: Reinforcement Learning with LTLf/LDLf restraining specifications. 2019

Linear Temporal Logic(LTL_f)

How to reward a robot that “eventually delivers coffee each time it gets a request”?

The LTL_f is the classical linear time logic interpreted over finite traces, formed by a finite sequence of propositional interpretations.

$$\varphi \quad ::= \quad \phi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \circ\varphi \mid \varphi_1 \,\mathcal{U}\, \varphi_2$$

“*all coffee requests from person p will eventually be served*”

$$\Box(request_p \rightarrow \Diamond coffee_p)$$

De Giacomo, Iocchi, Favorito and Patrizi,
Foundations for Restraining Bolts: Reinforcement Learning with LTLf/LDLf restraining specifications. 2019

Linear Dynamic Logic(LDL_f)

LDL_f is a proper extension of LTL_f.

Formally, **LDL_f** formulas φ are defined as follows:

$$\begin{array}{lcl} \varphi & ::= & tt \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \langle \varrho \rangle \varphi \\ \varrho & ::= & \phi \mid \varphi? \mid \varrho_1 + \varrho_2 \mid \varrho_1 ; \varrho_2 \mid \varrho^* \end{array}$$

"*all coffee requests from person p will eventually be served*"

$$[true^*](request_p \rightarrow < true^* > coffee_p)$$

NMRDP’s: Non-Markovian Reward Decision Process

$$\bar{M} =< S, A, Tr, \bar{R}>$$

- $\bar{R}$ new reward: real-valued function over infinite state-action sequences (since histories of state is infinite):
$$\bar{R}\colon (S \times A)^{*} \rightarrow \mathcal{R}$$

- *LTL$_f$/LDL$_f$* provides an intuitive language reward

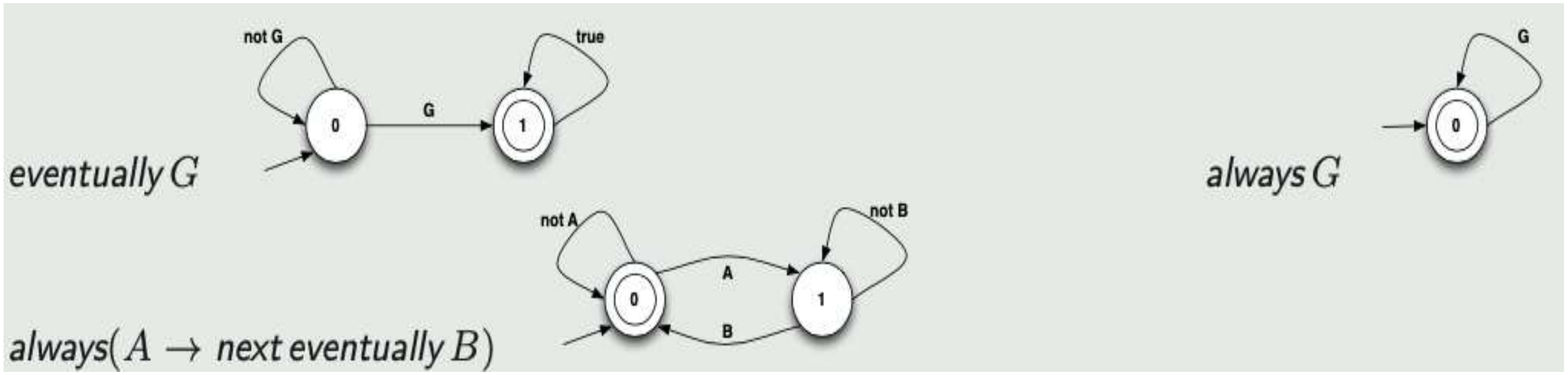
$$\downarrow$$

$\bar{R}$ can be espress as a set of pairs (φ_i, r_i)

De Giacomo, Iocchi, Favorito and Patrizi, Foundations for Restraining Bolts: Reinforcement Learning with LTLf/LDLf restraining specifications. 2019

Computing Deterministic Finite Automata (DFA) for LTL$_f$/LDL$_f$ formulas

- For any LTL_f/LDL_f formula φ_i we can build a DFA $A_{\varphi_i} = < 2^P, Q_i, q_{i0}, \delta_i, F_i >$ that tracks satisfaction of φ_i.
- A_φ accepts finite trace π iff satisfies φ_i.

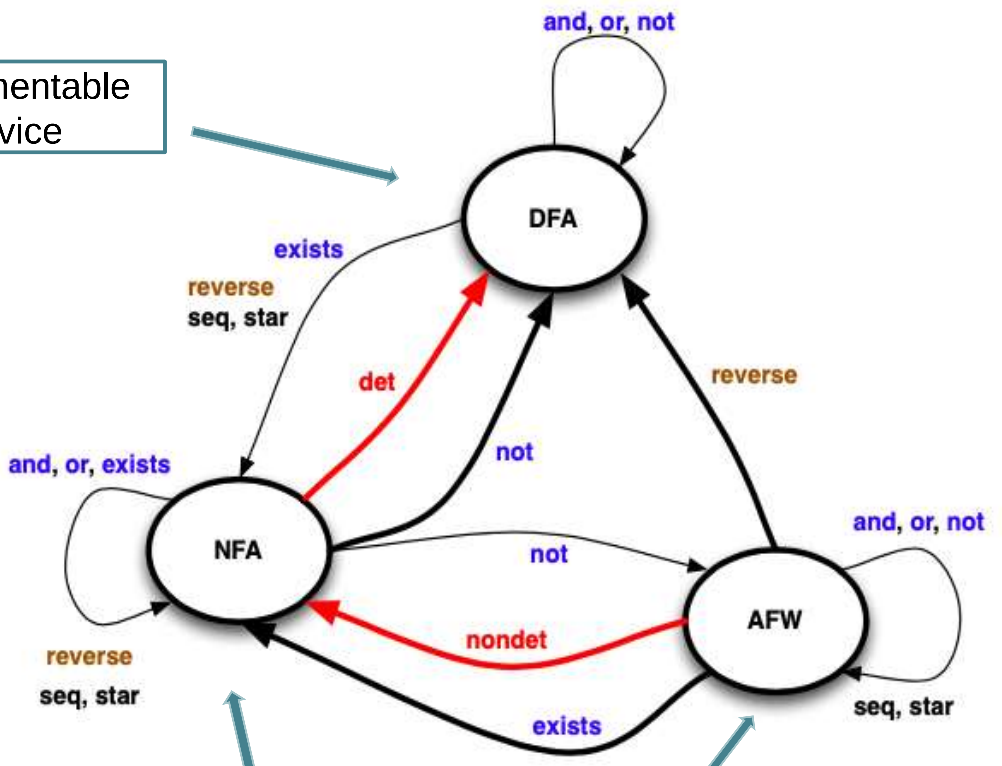


Giuseppe De Giacomo, Luca Iocchi, Fabio Patrizi "Decision making with temporal goals: Reinforcement Learning for Restrainig Bolts"

Brafman, De Giacomo and Patrizi , LTLf /LDLf non-markovian rewards. AAAI, 2018.

DFA (Deterministic Finite Automata), NFA (Non-Deterministic Finite Automata), AFW (Alternating Finite Automata on Words)

Implementable device

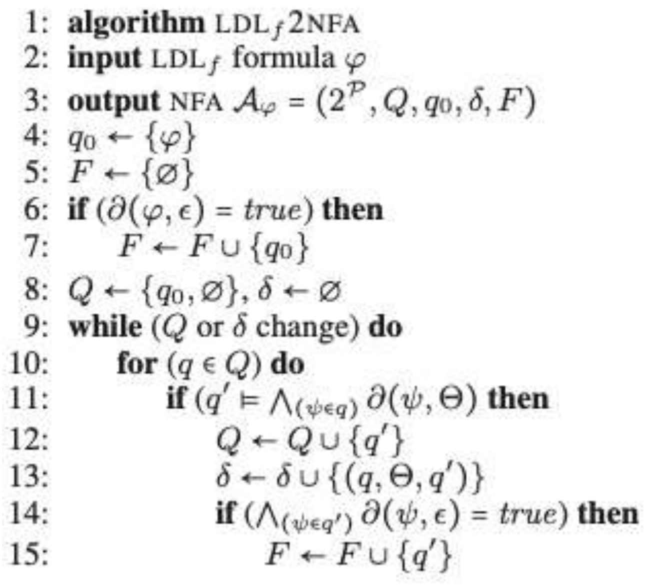


Mathematical devices

Giuseppe De Giacomo, Moshe Y. Vardi: Synthesis for LTL and LDL on Finite Traces

Giuseppe De Giacomo, Luca Iocchi, Fabio Patrizi "Decision making with temporal goals: Reinforcement Learning for Restrainig Bolts"

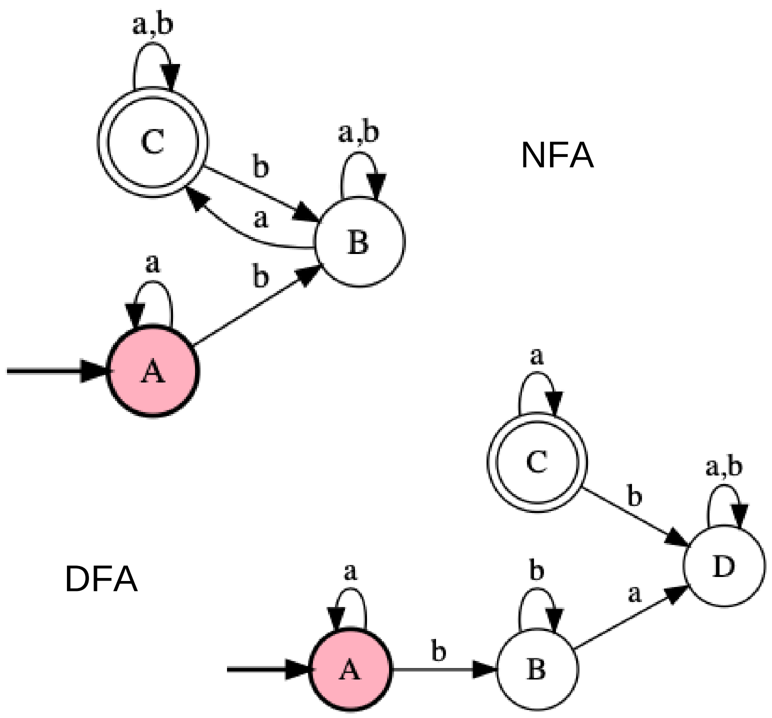
From NFA to DFA



```
algorithm LDL$_f$2NFA
input LDL$_f$ formula $\varphi$
output NFA $\mathcal{A}_\varphi = (2^{\mathcal{P}}, Q, q_0, \delta, F)$
$q_0 \leftarrow \{\varphi\}$
$F \leftarrow \{\varnothing\}$
if ($\partial(\varphi, \epsilon) = true$) then
    $F \leftarrow F \cup \{q_0\}$
$Q \leftarrow \{q_0, \varnothing\}$, $\delta \leftarrow \varnothing$
while ($Q$ or $\delta$ change) do
    for ($q \in Q$) do
        if ($q' \models \bigwedge_{(\psi \in q)} \partial(\psi, \Theta)$) then
            $Q \leftarrow Q \cup \{q'\}$
            $\delta \leftarrow \delta \cup \{(q, \Theta, q')\}$
            if ($\bigwedge_{(\psi \in q')} \partial(\psi, \epsilon) = true$) then
                $F \leftarrow F \cup \{q'\}$
```

Figure 1: LDL$_f$2NFA algorithm

Example:



❖ The NFA can be transformed into a DFA also *on-the-fly* (no construction of A_φ)

Brafman, De Giacomo and Patrizi , LTLf /LDLf non-markovian rewards. AAAI, 2018.

NMRDP with LTL_f / LDL_f rewards

DFA associated with $A_{\varphi_i} = < 2^P, Q_i, q_{i0}, \delta_i, F_i >$:

MDP $M' =< S', A, Tr', R' >$

- $S' = S \times Q_1 \times \ldots \times Q_m$.
- The reward is Markovian:

$$R'(q_{1,\ldots,}q_m, s, a, q'_{1,\ldots,}q'_m, s') = \sum_{i:q'_i \in F_i} r_i$$

Theorem 1:

The NMRDP $\bar{M} =< S, A, Tr, \bar{R}>$ is equivalent to the MDP $M' =< S', A, Tr', R' >$.

De Giacomo, Iocchi, Favorito and Patrizi, Foundations for Restraining Bolts: Reinforcement Learning with LTLf/LDLf restraining specifications. 2019

Another method: *Reward shaping*

The first three steps of this approach are the same of before for compilation of NMRDPs into an MDPs.

- Reward shaping is a well-known technique for MDPs that transforms the reward function in:

$$R'(s,a,s') = R'(s,a,s') + F(s,a,s')$$

$$\downarrow$$

shaping reward function

$$\downarrow$$

$$F(s,a,s') = \gamma\,\phi(s') - \phi(s)$$

(Optimality is preserved)

Non-Markovian Rewards Expressed in LTL: Guiding Search via Reward Shaping
Alberto Camacho,2 Oscar Chen,Scott Sanner,Sheila A. McIlraith

RL for NMRDP with LTL$_f$/LDL$_f$ rewards

NMRDP $M =< S, A, T_r, \{(\varphi_i, r_i)\}_{i=1}^{m} >$

$$\downarrow$$

MDP $M' =< S', A, Tr', R' >$

<u>Theorem 3</u>: *RL for LTL$_f$ / LDL$_f$ rewards over an NMRDP M with the properties expressed before, can be reduced to RL over the MDP $M' =< S', A, Tr', R' >$ with Tr' and R' hidden to the learning agent.*

De Giacomo, Iocchi, Favorito and Patrizi, Foundations for Restraining Bolts: Reinforcement Learning with LTLf/LDLf restraining specifications. 2019

RL with LTL$_f$ /LDL$_f$ restraining specifications

We are given:

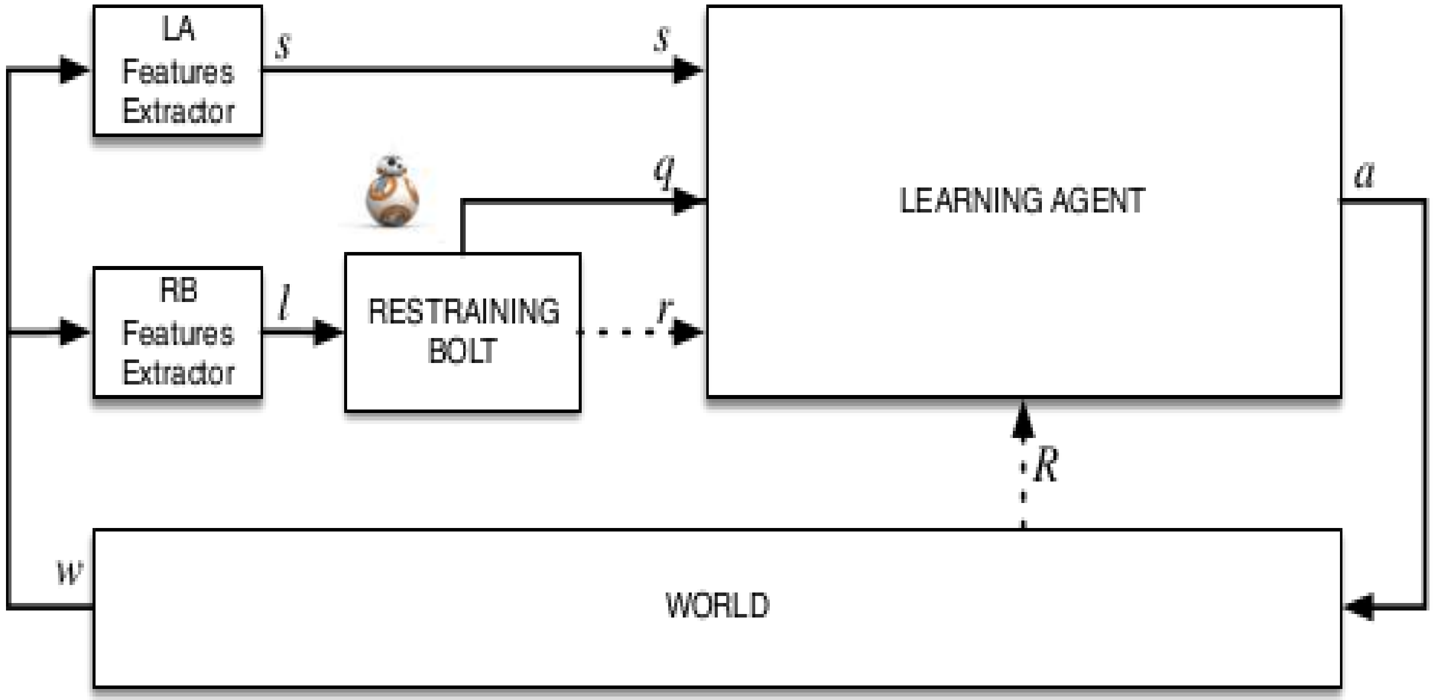
- A learning agent modeled by the MDP

$$M_{ag} = < S_{ag}, A_{ag}, Tr_{ag}, R_{ag} >$$

- $RB = < \mathcal{L}, \{(\varphi_i, r_i)\}_{i=1}^{m} >$ where:
  - $\mathcal{L}$ is the set of possible fluents' configurations
  - $\{(\varphi_i, r_i)\}_{i=1}^{m}$ is a set of restraining specifications with:
    - φ_i an LTL$_f$/LDL$_f$ formula
    - r_i the reward associated with φ_i

De Giacomo, Iocchi, Favorito and Patrizi, Foundations for Restraining Bolts: Reinforcement Learning with LTLf/LDLf restraining specifications. 2019

Learning Agent and Restraining Bolt



The agent receives **reward** based on R_{ag} and the pairs (φ_i, r_i)

Solution: $\bar{\rho} : (Q_1 \times \cdots \times Q_m \times S)^* \rightarrow A$

[De Giacomo, Iocchi, Favorito and Patrizi, Foundations for Restraining Bolts: Reinforcement Learning with LTLf/LDLf restraining specifications. 2019]

Main theorem(6)

Reinforcement Learning *with LTL$_f$ / LDL$_f$ restraining specifications* $M_{ag}^{rb} =< M_{ag,}RB >$ *with* $M_{ag} = < S_{ag}, A_{ag}, Tr_{ag}, R_{ag} >$ and $RB = < \mathcal{L}, \{(\varphi_i, r_i)\}_{i=1}^{m} >$ can be reduced to RL over the MDP

$$M_{ag}^{q} =< Q_1 \times \ldots \times Q_m \times S, A, T_r'', R_{ag}'' >$$

and optimal policies ρ_{ag}^{new} *for* M_{ag}^{rb} can be learned by learning corresponding optimal policies for M_{ag}^{q}.

De Giacomo, Iocchi, Favorito and Patrizi, Foundations for Restraining Bolts: Reinforcement Learning with LTLf/LDLf restraining specifications. 2019

Experimental Section

From:

Original Paper experiments:
Breakout,
Sapientino,
Cocktail Party,
Minecraft.

To:

Our implementations:
Chess Moves Learning
&
Pick and Place Robot (only sketched)

Experiments General Setup

Assumptions

1. The *simulator* is able to model all the relevant evolutions of the world
2. There are sensors for Learning Agent and Restraining Bolt

Episode reset/end conditions

1. *DFA* state satisfies formulas
2. *DFA* state is a fail
3. Maximum number of action reached

Algorithm Specifications

1. *SARSA* algorithm
2. The policy must maximizes the cumulative *reward*

An experiment from the paper: Sapientino

RL Agent

Can have two configurations:

1. OMNI: up, down, left, right actions
2. DIFFERENTIAL: forwards, backwards, turn left, turn right

The state representation is given by the agent pose: f_x, f_y, f_θ

So the agent just learn how to move through the grid

Restraining Bolt

Specification:

1. Visit at least two cells per color for each color
2. Visit all triplets of each color, given an order among colors

<<Beeps on visited colors and on actions>

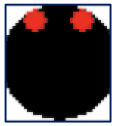
Specifications are expressed with LTL formulas as:

$$\neg bip\,\mathcal{U}(\textstyle\bigvee_{j=1,2,3} cell_{C_1,j} \wedge bip) \wedge$$
$$\textstyle\bigwedge_{j=1,2,3} \Box(cell_{C_1,j} \wedge bip \rightarrow \bigcirc\Box(bip \rightarrow \neg cell_{C_1,j})) \wedge$$
$$\textstyle\bigvee_{j=1,2,3} \Box(cell_{C_1,j} \wedge bip \rightarrow \bigcirc(\neg bip\,\mathcal{U} \bigvee_{k\neq j} cell_{C_1,k} \wedge bip)$$

Implementation: Chess Moves -1 (abstract)

RL Agent

Omnidirectional token robot that will learn different chess moves



The state representation is given by the agent pose f_x, f_y

The environment is a 5x7 grid

Restraining Bolt

Two important specifications:

1. Visit four cells for each color
2. Perform each chess move by respecting its particular order (not random moves)

<<Beeps on visited colors and on actions>>

Focus on the reward

The cumulative *R* will be the sum of goals and fails coming from both part (RL and RB)

GoalStep + RAGoal + RAFail

Implementation: Chess Moves -2 (code details)

Two python dictionary are implemented: **tokenBip** and **colorBip**

TOKENS = [[’name', color', x, y], [’t2', green', 1,1], [’t3', green', 2,2], [’t4', green', 3,3]…]

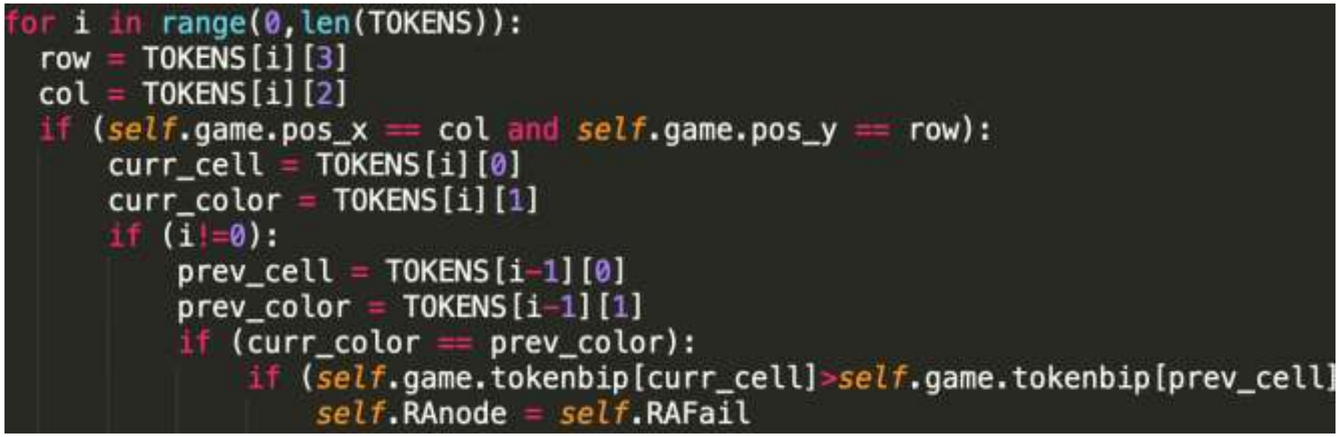
-Class RewardAutoma

Each method is finalized to perform what is described by LTL formulas, according to the Restraining Bolts specifications

Main method:

update

-check for color and moves orders
-update the current state with RB rewards



```
for i in range(0,len(TOKENS)):
  row = TOKENS[i][3]
  col = TOKENS[i][2]
  if (self.game.pos_x == col and self.game.pos_y == row):
      curr_cell = TOKENS[i][0]
      curr_color = TOKENS[i][1]
      if (i!=0):
          prev_cell = TOKENS[i-1][0]
          prev_color = TOKENS[i-1][1]
          if (curr_color == prev_color):
              if (self.game.tokenbip[curr_cell]>self.game.tokenbip[prev_cell]
                  self.RAnode = self.RAFail
```

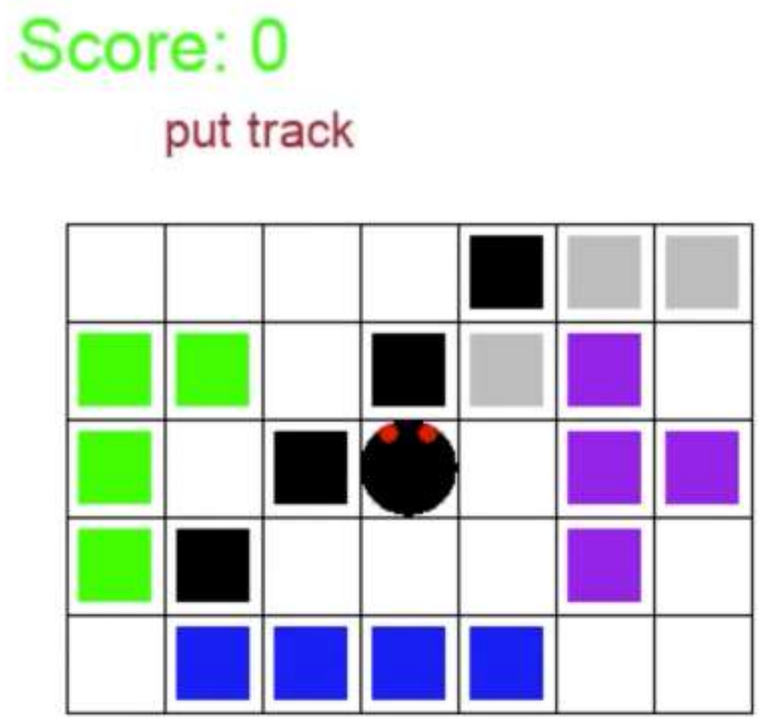
check the ordered motion of the robot

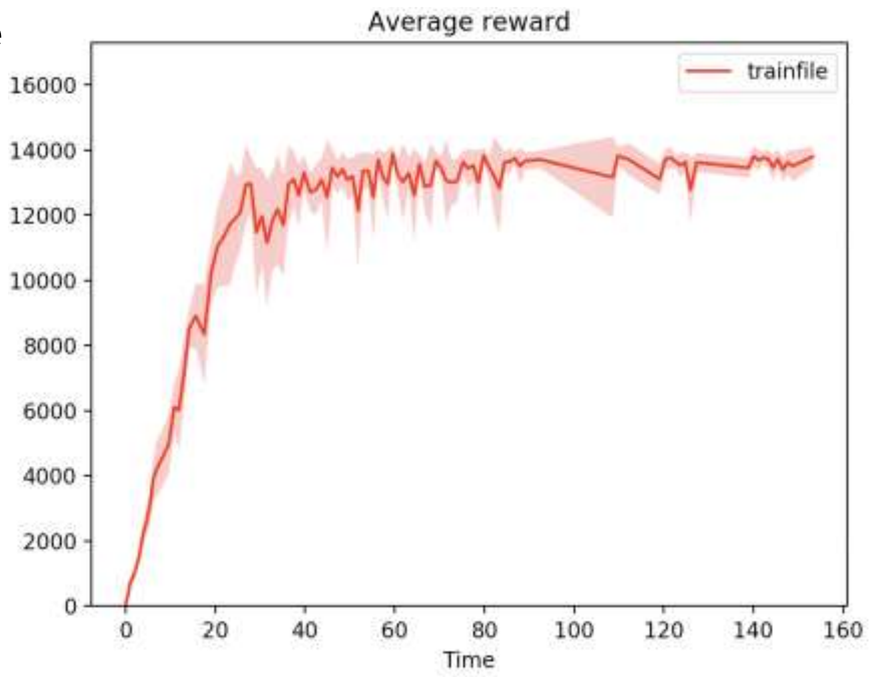
-Class Chess

Here we can found all we need to perform the RL implementation:
Update positions, actions and colors + Reset functions + Rendering

Implementation: Chess Moves -3 (results)

The agent has learned an **optimal policy** in a short time



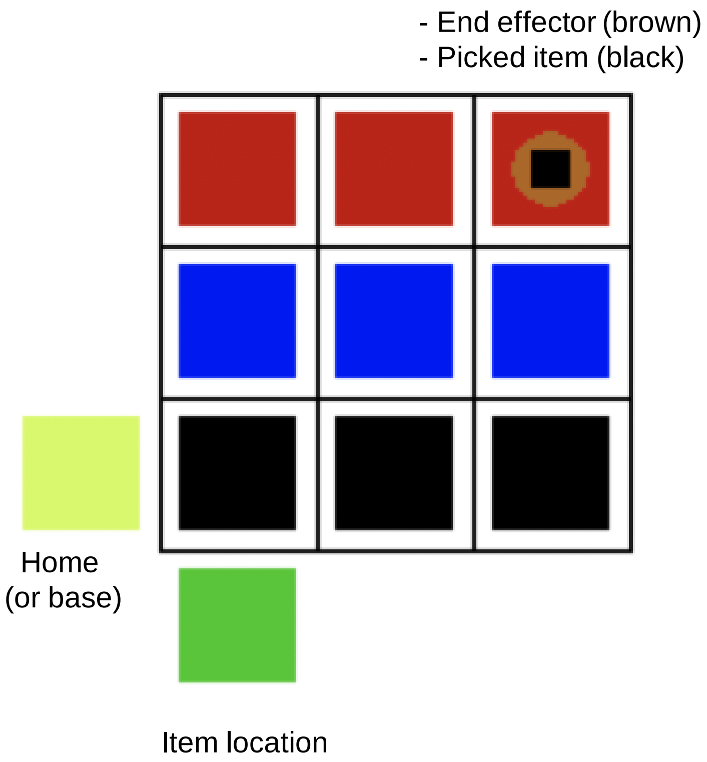


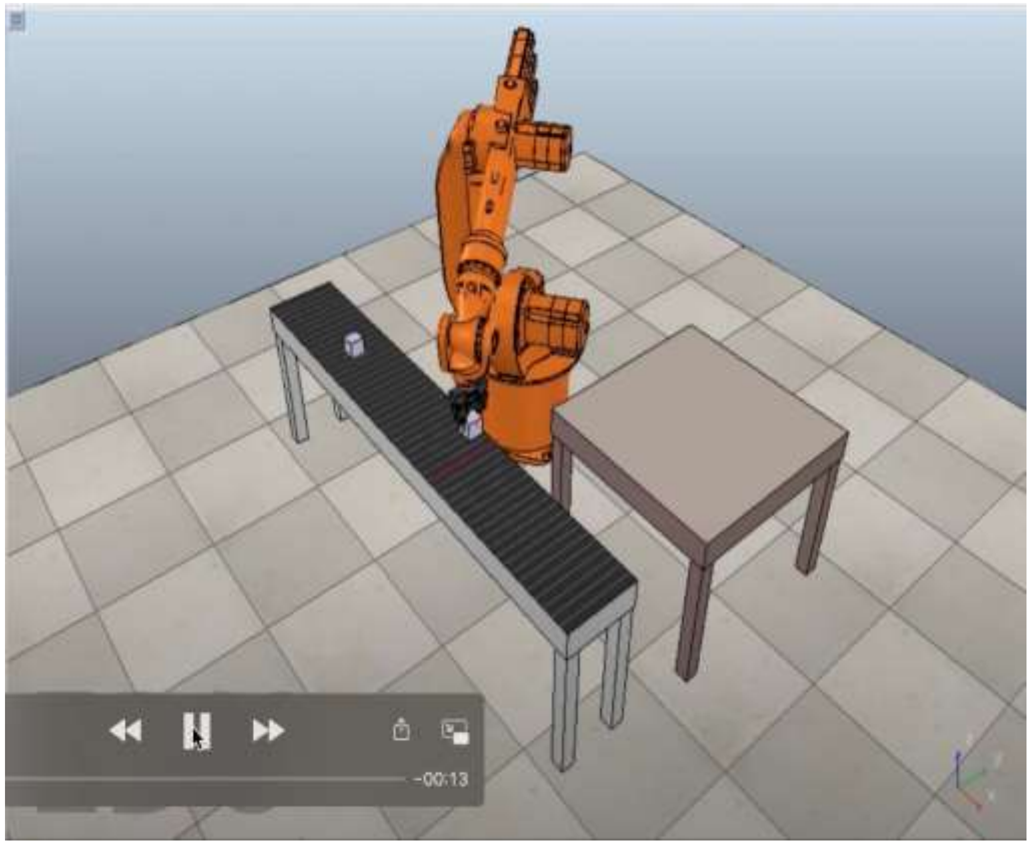
- Learning algorithm: SARSA
- Alpha: -0.99
- Epsilon: -2
- Gamma: 0.99
- n-steps: 20
- StopOnGoal: True
- MaxTime: 250

The robot computes in this sequence the task, keeping order

Knigth → Rock → King → Bishops → Queen

Pick and place task (still abstract)





RL agent and RB

RL Agent

The agent is a manipulator which must carry out the task of picking objects (once at time) from a conveyor belt and placing them on a table (actually, we have considered a relaxed problem).

The state representation is given by the position of the end-effector, the item location and if the end-effector is empty or not.

Restraining Bolts

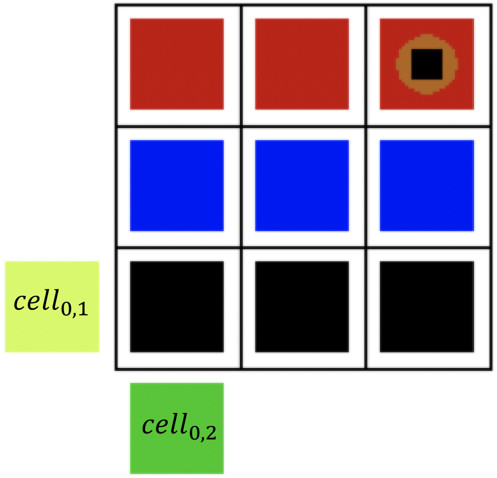
S1: place objects in a specific row (denoted by a color) following a specific order the grid is complete.

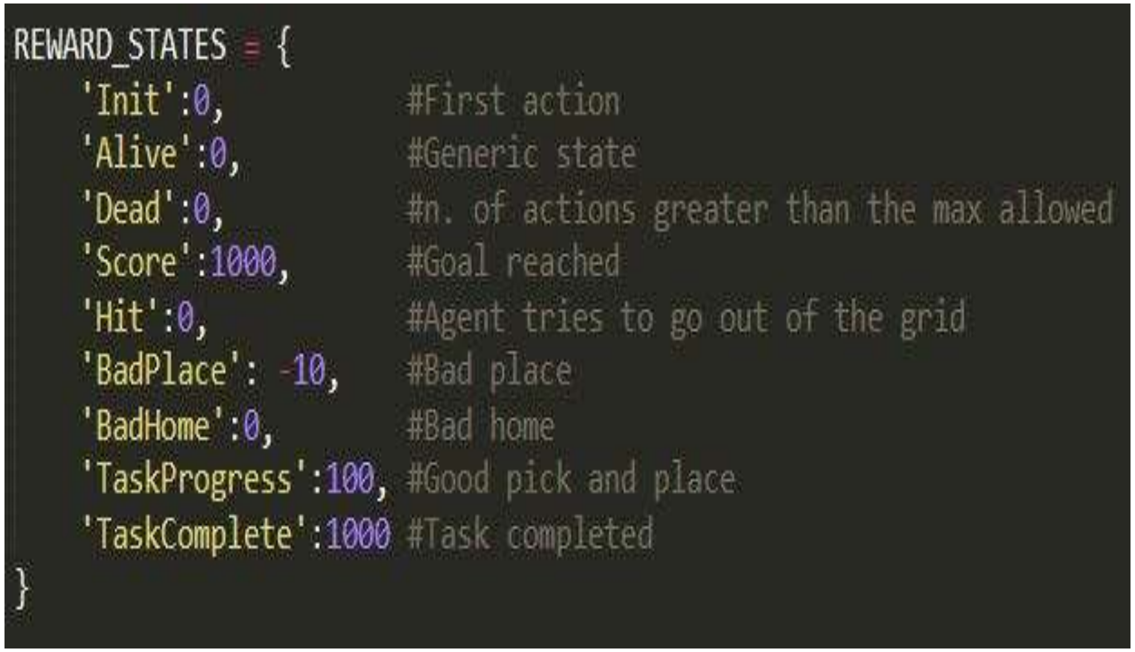
S2: place objects in a specific row (denoted by a color) in a random way the grid is complete.

The color of the current cell and the specific order are needed.

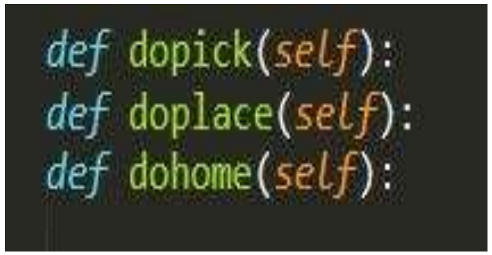
LTL_f pick and place formulas and code

- *S2*: $\square(ee \rightarrow cell_{0,1})\, U[(cell_{0,2} \wedge obj \wedge ee) \wedge \neg obj U \left(\vee_{j=1,2,3}\, cell_{c_{1,j}} \wedge \text{obj} \wedge ee\right) \wedge$
$\wedge_{j=1,2,3} \square(cell_{c_{1,j}} \wedge \text{obj} \rightarrow \text{o}\square(obj \rightarrow \neg cell_{c_{1,j}}))] \wedge$
$\vee_{j=1,2,3}\, \square(cell_{c_{1,j}} \wedge \text{obj} \rightarrow \text{o}(\neg obj U \vee_{j \neq k} cell_{c_{1,k}} \wedge obj))$





```
REWARD_STATES = {
    'Init':0,           #First action
    'Alive':0,          #Generic state
    'Dead':0,           #n. of actions greater than the max allowed
    'Score':1000,       #Goal reached
    'Hit':0,            #Agent tries to go out of the grid
    'BadPlace': -10,    #Bad place
    'BadHome':0,        #Bad home
    'TaskProgress':100, #Good pick and place
    'TaskComplete':1000 #Task completed
}
```



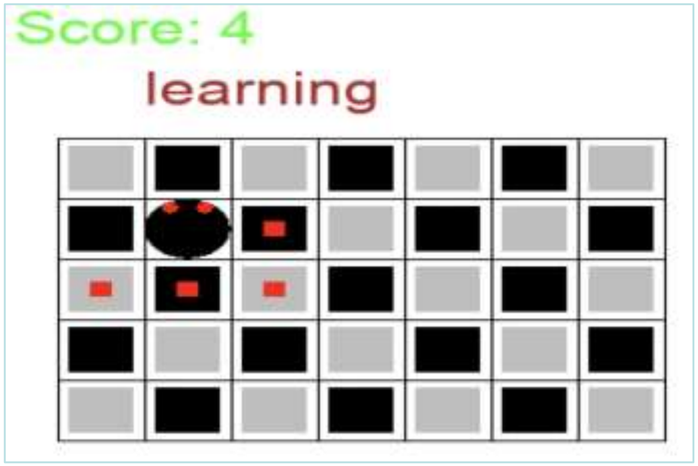
```
def dopick(self):
def doplace(self):
def dohome(self):
```

Conclusions

- It is possible to perform RL with LDL_f/LTL_f specification by using RL techniques on associated MDPs
- Agent as a black box
- Satisfaction of RB not guaranteed before nor after training (model-checking otherwise)
- Separation of concerns
  - feature spaces
  - modularity
  - reuse
  - simpler agents

Future works

- Implement a complete chess game using our agent and possibly with other learning agents



- Fully implementation of pick and place task
- Other examples (i.e.: consider an atonomous parking car)