Foundations for Restraining Bolts:
Reinforcement Learning with LTLf/LDLf Restraining Specifications

Giuseppe De Giacomo

Actions@KR18 – Oct. 29, 2018

Joint work with Marco Favorito, Luca Iocchi, & Fabio Patrizi
Restraining Bolts

RESTRAINING BOLT

A restraining bolt is a small cylindrical 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.

https://www.starwars.com/databank/restraining-bolt
Restraining Bolts

- **Restraining bolts** cannot rely on the internals of the agent they control.
- The controlled agent is **not built to be controlled by the restraining bolt**.
Two distinct representations of the world:
- one for the agent, by the designer of the agent
- one for the restraining bolt, by the authority imposing the bolt

Are these representations related to each other?
- NO: the agent designer and the authority imposing the bolt are not aligned  (why should they!)
- YES: the agent and the bolt act in the real world.

But can restraining bolt exist at all?
- YES: for example based on Reinforcement Learning!
RL with $\text{LTL}_f/\text{LDL}_f$ restraining bolt

Two distinct representations of the world $\mathcal{W}$:
- A learning agent represented by an MDP with **LA-accessible features** $S$, and reward $R$
- $\text{LTL}_f/\text{LDL}_f$ rewards $\{(\varphi_i, r_i)_{i=1}^m\}$ over a set of **RB-accessible features** $\mathcal{L}$

Solution: a non-Markovian policy $\rho : S^\ast \rightarrow A$ that is optimal wrt rewards $r_i$ and $R$.

*Observe $\mathcal{L}$ not used in $\rho$!*
Formally:

**Problem definition:** RL with LTL$_f$/LDL$_f$ restraining specifications

Given

- a **learning agent** $M = \langle S, A, Tr_{ag}, R_{ag} \rangle$ with $Tr_{ag}$ and $R_{ag}$ unknown, and
- a **restraining bolt** $RB = \langle \mathcal{L}, \{(\varphi_i, r_i)\}_{i=1}^m \rangle$ formed by a set of LTL$_f$/LDL$_f$ formulas $\varphi_i$ over $\mathcal{L}$ with associated rewards $r_i$.

Learn a non-Markovian policy $\rho : S^* \rightarrow A$ that maximizes the expected cumulative reward.
Example: **Breakout + remove column left to right**

- **Learning Agent**
  - **LA features**: paddle position, ball speed/position
  - **LA actions**: move the paddle
  - **LA rewards**: reward when a brick is hit

- **Restraining Bolt**
  - **RB features**: bricks status (broken/not broken)
  - **RB LTL$_f$/LDL$_f$ restraining specification**: all the bricks in column $i$ must be removed before completing any other column $j > i$ ($l_i$ means: the $i$th column of bricks has been removed):
    $$\langle (\neg l_0 \land \neg l_1 \land \ldots \land \neg l_n)^* ; (l_0 \land \neg l_1 \land \ldots \land \neg l_n) ; (l_0 \land \neg l_1 \land \ldots \land \neg l_n)^* ; \ldots ; (l_0 \land l_1 \land \ldots \land l_n) \rangle_{tt}$$
Example: **SAPIENTINO + pair colors in a given order**

- **Learning Agent**
  - **LA features**: robot position \((x, y)\) and facing \(\theta\)
  - **LA actions**: forward, backward, turn left, turn right, beep
  - **LA reward**: negative rewards are given when the agent exits the board.

- **Restraining Bolt**
  - **RB features**: color of current cell, just beeped
  - **RB LTL\(_f\)/LDL\(_f\) restraining specification**: visit (just beeped) at least two cells of the same color for each color, in a given order among the colors
Example: CocktailParty Robot + don’t serve twice & no alcohol to minors

- **Learning Agent**
  - **LA features**: robot’s pose, location of objects (drinks and snacks), and location of people
  - **LA actions**: move in the environment, can grasp and deliver items to people
  - **LA reward**: rewards when a deliver task is completed.

- **Restrainting Bolt**
  - **RB features**: identity, age and received items
  - **RB LTLf/LDLf restraining specification**: serve exactly one drink and one snack to every person, but do not serve alcoholic drinks to minors

*(in practice, tools like Microsoft Cognitive Services Face API can be integrated into the bolt to provide this information.)*
Building blocks

- **Classic Reinforcement Learning:**
  - An **agent** interacts with an **environment** by taking **actions** so to maximize **rewards**;
  - No knowledge about the transition model, but assume Markov property (history does not matter): Markov Decision Process (MDP)
  - Solution: **Markovian policy** $\rho : S \rightarrow A$

- **Temporal logic on finite traces** (De Giacomo, Vardi 2013):
  - Linear-time Temporal Logic on Finite Traces $\text{LTL}_f$
  - Linear-time Dynamic Logic on Finite Traces $\text{LDL}_f$
  - Reasoning: transform formulas $\varphi$ into NFA/DFA $A_\varphi$
    - s.t. for every trace $\pi$ and $\text{LTL}_f/\text{LDL}_f$ formula $\varphi$: $\pi \models \varphi \iff \pi \in \mathcal{L}(A_\varphi)$

- **RL for Non-Markovian Decision Process with $\text{LTL}_f/\text{LDL}_f$ rewards** (Brafman, De Giacomo, Patrizi 2018):
  - Rewards depend from history, not just the last transition;
  - Specify proper behaviours by using $\text{LTL}_f/\text{LDL}_f$ formulas;
  - Solution: **Non-Markovian policy** $\rho : S^* \rightarrow A$
  - Reduce the problem to MDP (with extended state space)
Lemma (BDP18): Every non-Markovian policy for $\mathcal{N}$ is equivalent to a Markovian policy for $\mathcal{M}$ which guarantees the same expected reward, and vice versa.

Theorem (BDP18): One can find optimal non-Markovian policies solving the $\mathcal{N}$ by searching for optimal Markovian policies for $\mathcal{M}$.

Corollary: We can reduce non-Markovian RL for $\mathcal{N}$ to standard RL for $\mathcal{M}$.
Problem definition: **RL with LTL\textsubscript{f}/LDL\textsubscript{f} restraining specifications** (De Giacomo, Favorito, Iocchi, Patrizi 2018)

Given

- a **learning agent** \( M = \langle S, A, Tr_{ag}, R_{ag} \rangle \) with \( Tr_{ag} \) and \( R_{ag} \) unknown, and
- a **restraining bolt** \( RB = \langle L, \{ (\varphi_i, r_i) \}_{i=1}^{m} \rangle \) formed by a set of LTL\textsubscript{f}/LDL\textsubscript{f} formulas \( \varphi_i \) over \( L \) with associated rewards \( r_i \).

learn a non-Markovian policy \( \rho : S^* \rightarrow A \) that maximizes the expected cumulative reward.

**Theorem** (De Giacomo, Favorito, Iocchi, Patrizi 2018)

**RL with LTL\textsubscript{f}/LDL\textsubscript{f} restraining specifications** for learning agent \( M = \langle S, A, Tr_{ag}, R_{ag} \rangle \) and restraining bolt \( RB = \langle L, \{ (\varphi_i, r_i) \}_{i=1}^{m} \rangle \)

- can be **reduced to classical RL over the MDP** \( M' = \langle Q_1 \times \cdots \times Q_m \times S, A, Tr'_{ag}, R'_{ag} \rangle \)
- i.e., the optimal policy \( \rho'_{ag} \) learned for \( M' \) corresponds to an optimal policy of the original problem.

\[
R'_{ag}(q_1, \ldots, q_m, s, a, q'_1, \ldots, q'_m, s') = \sum_{i: q'_i \in F_i} r_i + R_{ag}(s, a, s')
\]

We can rely on off-the-shelf RL algorithms (Q-Learning, Sarsa, ...)!
RL with $\text{LTL}_f/\text{LDL}_f$ restraining specifications (De Giacomo, Favorito, Iocchi, Patrizi 2018)

Our approach:

- Transform each $\varphi_i$ into DFA $\mathcal{A}_{\varphi_i}$
- Do RL over an MDP $\mathcal{M}'$ with a transformed state space:

$$S' = Q_1 \times \cdots \times Q_m \times S$$

Notice: the agent ignores RB features $\mathcal{L}$!

RL relies on standard algorithms (e.g. Sarsa($\lambda$))
Relationship between the LA and RB representations

**Question 1:** What is the relationship between $S$ and $L$ that needs to hold, in order to allow the agent to learn an optimal policy for the RB restraining specification?

**Answer:** *None!* The LA will learn anyway to comply as much as possible to the RB restraining specifications. *Note that from a KR viewpoint being able to synthesize policies by merging two formally unrelated representations $S$ for LA and $L$ for RB is unexpected, and speaks loudly about certain possibilities of RL vs. reasoning/planning.*

**Question 2:** Will LA policies surely satisfy RB restraining specification?

**Answer:** *Not necessarily!* “You can’t teach pigs to fly!” But if it does not then anyway no policy are possible!

*If we want to check formally that the optimal policy satisfies the RB restraining specification, we first need to model how LA actions affects RB $L$ (the glue) and then we can use e.g., model checking*

**Question 3:** Is the policy computed the same as if we did not make distinction between the features?

**Answer:** *No!* We learn optimal non-Markovian policies of the form $S^* \rightarrow A$ not of the form $(S \cup L)^* \rightarrow A$
Outlook

The idea of restraining bolt can be subscribed to that part of research generated by the urgency of providing safety guarantees to AI techniques based on learning.


However, the Restraining Bolt must impose its requirements without knowing the internals of controlled agent, which remains a black-box.