Perfect Observability is a Myth: Restraining Bolts in the Real World

Mudit Verma[1], **Naman Shah**[1], **Rashmeet Kaur Nayyar**[1], **and Akkamahadevi Hanni** [1],
[1] Arizona State University
{mverma13, namanshah, rmnayyar, ahanni}@asu.edu

Abstract

In the presence of a restraining bolt that models features of the world that are distinct from those modeled by a reinforcement learning agent, a fully observable setting is assumed for the developer of the restraining specifications. However, the real world is often noisy and partially observable. We consider the setting where the generator of the specification infers a noisy map of the features, thereby disentangling the bolt's observation from its feature space. A set of low and high probability fluents is extracted from bolt's observations which are used to specify the restrictions in linear temporal logic. We empirically evaluate our approach's performance for noisy restraining bolts with AI agents and show that the agent can still learn to effectively conform to the correct specifications through appropriate reward shaping.

1 Introduction

Recent years have seen a rise in learning-based agents that employ various techniques to enable them to learn behaviors that can be used to solve complex problems. While these techniques (Wiering and Van Otterlo 2012; Kaiser et al. 2019; Achiam et al. 2017; Gullapalli, Franklin, and Benbrahim 1994; Lin 1993) use acquired experience to learn policies that maximize an agent's utility, they fail to consider constraints or specifications imposed by external entities. For example, consider a hospital security guard working along with a guard robot. The security guard may be a human who has access to features of the world like the security clearance of a person. The guard robot might have been programmed to patrol the area and look for threats but may not have access to the security clearance of people working at the hospital. In this situation, the human guard may impose a constraint on the robot as *avoid performing safety checks on people with high-security clearance*. This is one of the examples representatives of scenarios with constraints enforced by external entities.

Multiple approaches (Littman 2015; Littman et al. 2017; De Giacomo and Vardi 2013; De Giacomo and Rubin 2018; De Giacomo et al. 2019, 2020) have attempted to solve such problems by incorporating a human, or the restraining bolt - an entity who restrains the agent's behavior - a specification which translates into rewards for the agent. De Giacomo et al. (2020) proposes an approach that uses LTL_f (and LDL_f) formulas to encode constraints. LTL_f formulas encode reward functions using a deterministic finite-state automaton (*DFA*). Features used to define states for the *DFA* do not necessarily share environment features with the agent and can be designed independently. While this approach is able to impose such constraints, it assumes that the restraining bolt can deterministically infer the current state of the *DFA*. Following our example of security guard and robot, the guard might be using camera feeds in determining who the person is and thereby determining their security clearance. Such an inference can be noisy due to various reasons like the image observation is noisy, there are occlusions in the image, perception disabilities of the restraining bolt, etc. Such a scenario would not be handled by existing approaches like (De Giacomo et al. 2019). Another drawback of (De Giacomo et al. 2019) is that it only works with a factored state space for both the agent and the restraining bolt. Many problems may not have a factored state space and

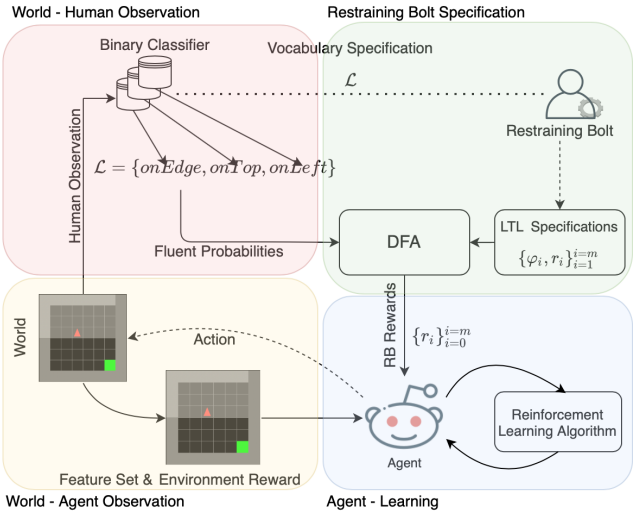

Figure 1: Overall Flow: The agent receives a feature set and environment reward from the world. The restraining bolt receives a noisy observation of the world, and the presence of fluents is estimated using binary classifiers. The restraining specifications over these fluents represented as LTLs are converted to DFAs. The fluent probabilities from classifiers and satisfaction of the state in DFAs are then used to shape the rewards for the learning agent.

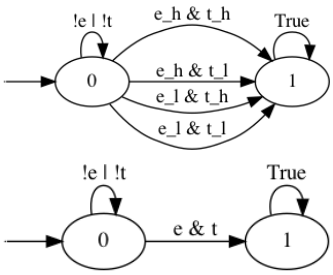

Figure 2: (a) Top: Augmented DFA with decomposed fluents e and l. The number of edges between state 0 and 1 are $2^{n_e} = 4$. (b) Bottom: Original DFA for the LTL: Eventually($e\&t$). Note that the number of edges between state 0 and 1 is 1.

rather have an image state observation for the agent. Moreover, we believe that handling noisy mappings from human observation in the advice framework is a novel effort since most of the works in this direction assume a symbolic mapping of the environment. We argue that perfect classifiers are a myth and handling noisy mappings from observation to fluents must be handled in the advice framework.

In this paper, we present an approach that not only allows for an image based state input for the agent, but also relaxes the assumption by De Giacomo et al. (2019) that a Restraining Bolt has perfect knowledge of fluents of the world. That is, our approach is capable of accepting noisy observations for the restraining bolt. Given that the sensors are noisy, the restraining bolt observes a probability distribution over propositions over their observation. This requires augmenting the *DFA* induced by the bolt's constraints to handle different levels of certainty over the observed state variables. Our approach introduces *low* and *high* confidence over observed propositions and uses these confidence levels to provide additional rewards. Although a naive augmentation to DFA can cause the number of transition edges to explode exponentially in the number of fluents, we propose an algorithm to achieve the same effect as the augmented DFA without modifying it. Similar to the previous work (De Giacomo et al. 2020), our approach does not assume similar vocabularies for the agent and the restraining bolt. Contrary to the current work, our approach handles noisy sensor models for the restraining bolt, and its representation is independent of the representation of the agent's state space. Our method has the ability to work with different representations such as factored, relational, and images. Finally, De Giacomo et al. (2019) requires that a restraining bolt has a domain model of the world in their vocabulary, which implies that for specifying even a simple constraint, the complete equivalent domain model must be obtained, which may contain many unnecessary vocabulary terms. This work, on the contrary, only requires vocabulary fluents for which a constraint has to be specified.

The remaining of the paper is structured as follows: section 2 details the approach used by the presented work, and section 3 presents the results for the empirical evaluation of our approach.

Algorithm 1: Transitioning over augmented DFA

Result: Reward, Next state over augmented DFA
Input: Original DFA D, Current DFA state q, Probabilities of Fluents C ;
if *C(e)* ≥ 0.5 **then**
  Truth(e) $= T$
else
  Truth(e) $= F$
end
E_list = D.get_edges(q) ;
E = e in E_list s.t. $\delta(q, Truth, e) = 1$ *;*
initialize $r = 0$ *;*
for *symbols in e* **do**
  if *0.5* $\leq$ *C(symbol)* $\leq$ *0.9* **then**
    r = r - d.LowConfidenceCost ;
  end
end
if *e is a terminal state* **then**
  r = r + d.TerminalReward
end
return r, e ;

2 Framework & Approach

2.1 Setting

The aim of De Giacomo et al. (2019) is to incorporate restraining specifications to influence the final policy of the agent. Hence, their setting involves a world $\mathcal{W}$ where an agent A_{ag} is acting by taking actions form the action set $\mathcal{A}$. However, the agent's sensors allow it to perceive the state $s \in \mathcal{S}$ such that s is derived from $\mathcal{W}$. Another entity, a Restraining Bolt perceives the same world as $l \in \mathcal{L}$ which, in turn, is derived from $\mathcal{W}$. A_{ag} models its task as an MDP $M_{ag} =< \mathcal{S}, \mathcal{A}, \mathcal{T}, R >$ where $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the transition function and R is the reward given by the environment. A restraining bolt also specifies a vocabulary of fluents $\mathcal{F}$ and a set of m LTL specifications φ with restraining rewards r as the set $\{\varphi_i, r_i\}_{i=0}^{i=m}$. The main result, Theorem 6 of (De Giacomo et al. 2019) states that this is a Non-Markovian Reward Decision Process (NMRDP) problem and can be reduced to an equivalent MDP $M =< \mathcal{S}', \mathcal{A}, \mathcal{T}', \mathcal{R}' >$ where $\mathcal{S}' = \mathcal{S} \times \mathcal{Q}_1 \times \mathcal{Q}_2 .. \times \mathcal{Q}_m$, where $\mathcal{Q}_i$ is the set of states of an equivalent Deterministic Finite State automata (DFA) $\mathcal{A}_{\varphi_i}$ of the LTL formula φ_i. Similarly, the same set of actions $\mathcal{A}$ is used and $\mathcal{T}'$ is the transition function defined as $\mathcal{S}' \times \mathcal{A} \times \mathcal{S}' \rightarrow [0, 1]$. The reward $\mathcal{R}'(s, q_1, q_2..q_m, a, s', q_1', q_2'..q_m') = \sum_{i:q_i \in F_i} r_i + R(s, a, s')$, where F_i is the set of terminal states in $\mathcal{A}_{\varphi_i}$.

2.2 Proposed Work

To allow our approach to be invariant to the representation of the agent's state space, we use a deep neural network to encode the state space of the agent and learn a probabilistic policy $\pi : \mathcal{S}' \times \mathcal{A} \rightarrow [0, 1]$. We decompose π as $\pi(s') = H(G(s)|Q)$ where $s' =< s, Q >$, function G represents a set of convolutional layers that act as a feature extractor, and H represents a set of dense layers that are responsible for predicting the action distribution. | is a con-

catenation operation, and Q is the concatenation of all the one-hot vectors for each *DFA* state q_i in each *DFA*.

We do not require a complete domain model in the fluent vocabulary $\mathcal{L}$. Rather, we allow a human observation of the world state w as h_l which in turn is approximated into fluents in the set $\mathcal{L}$. This approximation is done by any black-box classification algorithm $C(h_l) \times [0, 1]$ returning the probability of the fluent being true. Note that, we have a binary classifier for each fluent taking the input h_l which can either be a factored or an image-based representation of the world. This is where we capture the possibility of noise in the human observation since the classifiers can be noisy. This allows the agent to not only adhere to the specified constraints but also enable the bolt in the loop to observe it.

We propose algorithm 1 to incorporate the notion that the agent should comply with the constraints in a manner that is observable by the bolt. Figure 2 (b) shows the *DFA* for the *LTL* formula *"F(e & t)"* which translates to "Eventually the proposition 'e&t' is True". We decompose the fluent e as e_h, e_l, and $!e$ s.t. $e_h(e_l)$ represents that the bolt is able to infer the presence of the fluent with high(low) confidence and $!e$ represents that the fluent is absent. Out approach determines the truth value of these fluents using the classification score predicted by the classfier as follows:

$$e = \begin{cases} \mathrm{e}_h & C(e) > 0.9 \\ \mathrm{e}_l & 0.5 \leq C(e) \leq 0.9 \\ !\mathrm{e} & C(e) < 0.5 \end{cases} \quad (1)$$

By decomposing the fluent in this manner, we are able to track not only the attainment of the fluent e but also the confidence with which it is achieved (i.e. high or low). One possible way to achieve this is to modify the edges of the *DFA* to track the high and low confidence fluents. Fig. 2 (a) expresses a possible equivalent *DFA* that is obtained by decomposing e. This means that each edge in the original *DFA* fig. 2(b) with n_e fluents (without negation) would have 2^{n_e} number of edges in the corresponding *DFA* fig. 2(a).

We now discuss how such a decomposition helps in shaping the rewards for the agent and how does the Alg. 1 bypass the need for modifying the *DFA*.

In fig. 2(b), a transition $\delta(q_i, l, q_j)$ happens from the state q_i when proposition l (composed of fluents in $\mathcal{L}$) is evaluated as True. Fig. 2(a) extends this idea to instead make a transition over the decomposed variant of the fluent e. Hence, the fluent set $\mathcal{L}$ is augmented with the decomposed fluents as per eq. 1. Further, we augment the reward function $r(q_k)$ for the *DFA* to impart a negative reward whenever a transition is made over an edge l, i.e. whenever the presence of fluents is known with low confidence to the restraining bolt. Note that, even though l may consist of fluents known with low confidence, our method still allows the transition from q_i to q_j, albeit with a negative reward. The choice of this negative reward is dependent on the terminal reward of the *DFA* as chosen by the Restraining Bolt. This negative reward is often set to a value less than the terminal reward so that the transition eventually provides a smaller but positive additional reward.

Finally, as pointed out, the transformation of the DFA from fig. 2(b) to the DFA in fig. 2(a) will require adding exponential number of edges. This will significantly hamper the DFA transition computation. Our proposed Alg. 1 provides the intended effect of augmented DFA in fig. 2(a) without any modifications to the original DFA. The algorithm runs in two phases; in the first phase, it selects a transition based on the truth value of fluents determined using a threshold probability of 0.5. The second phase uses the confidence values of e_l (low confidence) fluents to provide a negative reward. Note that, as a simple extension to this work, rewards imparted for transitioning over low confidence fluents can easily be modified to depend upon the probability $P(e)$ rather than giving a fixed reward for each low confidence fluent.

3 Empirical Evaluation

While our approach is invariant to the agent's state space's representation, we implement our approach with image-based observations for both the agent and the restraining bolt. Our empirical evaluation is aimed at evaluating: I) What behaviors does the agent learn in a setting where the restraining bolt's observations are noisy? II) Can a visual observation be used to learn constraints specified by the restraining bolt? We implement our approach using three *RL* algorithms: *asynchronous actor critic (A2C)* (Mnih et al. 2016), *proximal policy optimization (PPO)* (Schulman et al. 2017), and *deep Q-networks (DQN)* (Mnih et al. 2015). Implementations of *PPO* and *A2C* were used from the repository https://github.com/lcswillems/torch-ac. We update the input embedding of these networks (after the feature extraction layers) to include the *DFA's* state. To minimize the information loss, we append the *DFA*-state after the convolutional layers which compute latent representations for the images inspired by the literature in the field of natural language processing. We evaluate our approach in *two* domains with a total of *five* constraints. Source code of our implementation is available at https://bit.ly/3snrh0H.

3.1 Gridworld

The first domain has a *6x6* grid where the goal of the agent is to reach a location marked by the green square, as shown in 3. The actions available to the agent are move forward, turn left or right in-place, and no-op. The agent receives a reward of -1.5 for each step and a reward of $+100$ for the terminal state where it reaches the green square. We use a total of *four* constraints for the agent in this setting. Table 1 specifies the exact *LTL* formulas used to specify the constraints. These *LTL* formulas encode the following constraints: i) Reach a state where the agent is on the edge and continue to be on the edge. ii) Avoid the top-right corner. iii) Visit the top-right corner only once. iv) Visit either the top-right corner or the bottom-left corner (not both) only once. We collect positive and negative samples from bolt's-observation space to learn binary classifiers for the fluents $\langle$ onTopRightCorner, onBottomLeftCorner, pointingUp, pointingDown, pointingLeft, pointingRight, onEdge $\rangle$. While the domain appears to be simple, the noncontinuous image-based state representation makes it difficult for the algorithms to learn due to lack of gradients for back-propagation.

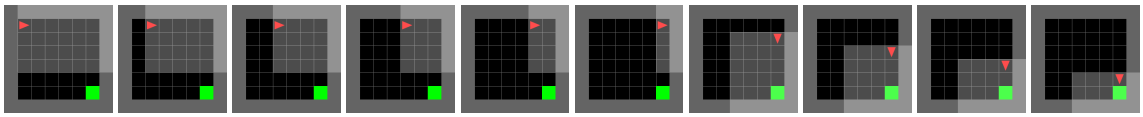

Figure 3: Execution of the policy learned with constraint provided by the restraining bolt requiring the agent to move only on the edges of the environment.

LTL specification	Algorithm	R^c_{max}	R_π	N_{steps}	N_{frames}	Constraint followed
`G(o)`	PPO	94	79.48	90	25k	✓
	A2C	94	63.64	75	80k	✓
	DQN	92	74.42	82	4k	✓
`!F(t)`	A2C	82	80.71	60	50k	✓
	DQN	82	79	59	4k	✓
`F(t and X(!(F(t))))`	PPO	92	77.78	87.13	133k	✓
	A2C	92	74.54	104.2	165k	✓
`(F(t and X(!(F(t))))) and !F(b))` `or (F(b and X(!(F(b))))) and !F(t))`	A2C	92	78.42	78	20k	✓
`F(c)`	DQN	—	-240	2900	2.8m	×

Table 1: Results for Gridworld domain where e = on_edge, t = top_right_corner, b = bottom_left_corner, and u = facing_upwards, and Breakout domain where c = brick_clear_left. Here, e, t, b, u, and c are fluents used to specify the LTL specifications.

3.2 Breakout

The second domain we chose to evaluate our system is the Atari Breakout. The goal of the agent is to break all the bricks using a ball. Actions available to the agent are to move the pedal left and right and to fire in the left, up, and right directions. The agent receives a reward of $+1$ for breaking each brick in the environment. An episode ends when the pedal misses to hit the ball. The table 1 specifies the constraints provided by the restraining bolt. These constraints enforce the agent to break the left column of bricks eventually. Unfortunately, the DQN agent was not able to solve the domain, and instead obtained a high negative reward. We intend to further test the method with other RL algorithms on Atari to substantiate results.

Analysis of the results: Table 1 provides the detailed results for our experiments. The table includes *LTL* formulas for the constraints, the *RL* algorithm used, the total possible reward with the constraint (R^c_{max}), the average reward achieved by the agent using the policy learned through our approach (R_π), the average number of steps taken by the agent to reach the goal (N_{steps}), the number of frames used to learn the final policy (N_{frames}), and boolean value denoting whether the constraint was followed or not. The first two LTLs test whether a specification as advice (positive reinforcement) and as a constraint (negative reinforcement) were correctly learned by the agent. The third LTL verifies whether this work can actually exercise temporal reward specification like agent must only visit the top right corner only once. The fourth LTL formula uses multiple fluents along with a temporal reward. Finally, we also test out our method on Atari-Breakout. The average return achieved by the learned policies for the Gridworld clearly show that the agent was able to reach the target goal whenever the constraint was provided while also complying with the constraints. Fig. 3 shows an execution of the policy where the agent was constrained to move only on the edges of the environment. The policy synthesized by the agent clearly shows that the agent complied with the constraint and continued to stay on edge. Inspection of case 4 : In LTL 4, the classifier for "b" fluent, at best, gave the probability 0.87; hence it was always either b_l or $!b$. Moreover, the fluent t worked perfectly over the grid world. As expected, the agent avoided the bottom left corner and chose to detour to the goal via top right corner, even though the DFA terminal rewards for a detour via either of the corners were the same. This implies that when the constraint specification is same for either going from top-right or bottom-left corner, since observation of bottom-left corner is noisy, the agent made pro-active efforts to instead detour via top-right corner.

Additional videos of obtained policies can be found along with the code. The verification of whether the advice was followed was done by manual inspection by the authors.

4 Conclusions

In this work, we extend the restraining bolt framework by De Giacomo et al. (2019) to allow for an image-based state space for the agent as well as an image-based noisy observation space for the restraining bolt. We relax the assumptions made by the previous work and propose an algorithm to shape the rewards by accounting for the noisy observations of fluents obtained by the restraining bolt. We evaluate our approach over multiple fluents in a Gridworld and an Atari Breakout setting. Our method accepts any LTL specification for all the OpenAI gym environments and our implementation is much more customizable and modular than previous work. Additionally, we experienced that shaping the LTL rewards is a complicated task (Şimşek and Barto 2006) and deserves further attention. Future work in this direction would require rigorous evaluation with more complex domains and other reinforcement learning algorithms.

References

Achiam, J.; Held, D.; Tamar, A.; and Abbeel, P. 2017. Constrained policy optimization. In *International Conference on Machine Learning*, 22–31. PMLR.

De Giacomo, G.; Iocchi, L.; Favorito, M.; and Patrizi, F. 2019. Foundations for restraining bolts: Reinforcement learning with LTLf/LDLf restraining specifications. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, 128–136.

De Giacomo, G.; Iocchi, L.; Favorito, M.; and Patrizi, F. 2020. Restraining bolts for Reinforcement Learning agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 13659–13662.

De Giacomo, G.; and Rubin, S. 2018. Automata-Theoretic Foundations of FOND Planning for LTLf and LDLf Goals. In *IJCAI*, 4729–4735.

De Giacomo, G.; and Vardi, M. Y. 2013. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI'13 Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, 854–860. Association for Computing Machinery.

Gullapalli, V.; Franklin, J. A.; and Benbrahim, H. 1994. Acquiring robot skills via reinforcement learning. *IEEE Control Systems Magazine* 14(1): 13–24.

Kaiser, L.; Babaeizadeh, M.; Milos, P.; Osinski, B.; Campbell, R. H.; Czechowski, K.; Erhan, D.; Finn, C.; Kozakowski, P.; Levine, S.; et al. 2019. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374* .

Lin, L.-J. 1993. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science.

Littman, M. 2015. Programming agent via rewards. *Invited talk at IJCAI* .

Littman, M. L.; Topcu, U.; Fu, J.; Isbell, C.; Wen, M.; and MacGlashan, J. 2017. Environment-independent task specifications via GLTL. *arXiv preprint arXiv:1704.04341* .

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, 1928–1937. PMLR.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature* 518(7540): 529–533.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* .

Şimşek, Ö.; and Barto, A. G. 2006. An intrinsic reward mechanism for efficient exploration. In *Proceedings of the 23rd international conference on Machine learning*, 833–840.

Wiering, M.; and Van Otterlo, M. 2012. Reinforcement learning. *Adaptation, learning, and optimization* 12(3).