

Neuro-Symbolic Injection of LTLf Constraints in Autoregressive Reinforcement Learning Policies

Ashkan Ansarifard, Matteo Mancanelli, Elena Umili and Fabio Patrizi

University of Rome La Sapienza, Rome, Italy

Abstract

In this work we study offline reinforcement learning (RL) under temporally extended task constraints expressed in Linear Temporal Logic over finite traces (LTL_f). Recently, transformer-based approaches such as Trajectory Transformers and Decision Transformers have been adopted to address RL as a sequence modeling problem. However, these methods optimize purely for reward and do not account for high-level temporal requirements. Here, we introduce a neurosymbolic framework that injects LTL_f background knowledge into such transformer-based RL policies. Our approach compiles LTL_f formulas into deterministic finite automata (DFAs) and integrates them into the learning process through a differentiable representation and a logic-based loss function. In particular, we derive differentiable satisfaction signals from DFA progression and use them as a regularization term during training. The resulting method is architecture-agnostic across different models. We evaluate the proposed framework on navigation environments with specification suites covering combinations of safety and reachability temporal properties. Experimental results show that incorporating background knowledge not only improves constraint satisfaction, but also maintains competitive return compared to vanilla baselines.

Keywords

Safe Reinforcement Learning, Neurosymbolic Knowledge Injection, Linear Temporal Logic

1. Introduction

Safety is a central concern in reinforcement learning (RL) [1], particularly when agents are deployed in real-world or safety-critical settings such as robotics, autonomous navigation, and decision-support systems, where undesirable behaviors may incur irreversible damage, financial loss, or human harm [2]. Ensuring that learned policies respect some user-defined constraints is therefore crucial for increasing the system correctness and reliability. This challenge becomes even more pronounced in offline RL [3], where policies are learned exclusively from pre-collected data and cannot rely on online interaction to detect and correct unsafe behavior.

Safe decision making often requires constraints that are *temporally extended*, rather than purely local or instantaneous [4]. Many requirements cannot be verified from a single state-action pair, but only over entire trajectories. For instance, consider the specification: *"eventually reach the goal while always avoiding hazards"*. This requirement combines two temporal modalities: (i) a *liveness* condition (the goal must be reached at some point before the episode ends), and (ii) a *safety* condition (no hazardous state may be visited at any preceding time step). A policy that briefly steps on a hazard—even if it later reaches the goal—violates the specification. A policy that remains safe but never reaches the goal also fails. Satisfaction therefore depends on the *entire trace*, not on local reward maximization alone.

In *offline reinforcement learning* (RL), enforcing such constraints is particularly challenging. Since no online interaction is allowed, constraint violations cannot be corrected via trial-and-error exploration. Sequence-model-based policies—such as Trajectory Transformers (TTs) [5] and Decision Transformers (DTs) [6]—are appealing in this setting because they generate trajectories autoregressively and naturally model long-range dependencies. However, they operate as probabilistic sequence generators, while

Joint Workshop on Statistics and Knowledge Integration for Logic, Learning, Ethical Decisions, and LLMs (SKILLED-LLMs 2026), co-located with FLoC 2026, Lisbon, Portugal

✉ ansarifard.1970082@studenti.uniroma1.it (A. Ansarifard); mancanelli@diag.uniroma1.it (M. Mancanelli); umili@diag.uniroma1.it (E. Umili); patrizi@diag.uniroma1.it (F. Patrizi)

🆔 0009-0005-9247-1786 (A. Ansarifard); 0009-0004-9547-7115 (M. Mancanelli); 0000-0002-5639-6038 (E. Umili); 0000-0002-9116-251X (F. Patrizi)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

task and safety requirements are typically expressed as formal logical specifications. This creates a gap between generative modeling and symbolic correctness guarantees.

In this paper, we study *symbolic constraint injection* [7, 8] for offline transformer policies using Linear Temporal Logic over finite traces (LTL_f) [9] as specification language. LTL_f provides a natural language for specifying unambiguous temporal requirements over episodic tasks. We compile LTL_f formulas into deterministic finite automata (DFAs) and integrate the resulting automata into the policy pipeline by training-time regularization, via differentiable soft-satisfaction signals derived from automaton progression. Our framework is architecture-agnostic across autoregressive sequence models that expose a differentiable distribution over the symbols consumed by the DFA. The logic loss defined using the DFA only requires differentiable symbol assignments at each step of the trace, from which the satisfaction signal are computed. We show how our framework can support different transformer-based RL architectures, like TT and DT, through a shared interface that abstracts away architectural differences. It would be possible also to extend our approach with test-time automaton-aware constrained decoding, where generation is restricted to action sequences consistent with the logical specification.

We evaluate the proposed approach on temporally extended tasks in a navigation environment, ColourBomb [4]. The considered specifications span invariant safety, reachability, and combined reach-while-safe objectives. The results section is structured to systematically expose the behavior of the method: the effects of logic regularization in TT and DT, cross-benchmark trade-offs between return and logical satisfaction, and runtime overhead. To summarize, we make the following contributions:

- A unified neurosymbolic offline sequence-RL framework for injecting LTL_f constraints into transformer-based policies through automata-based modules.
- A canonical finite-trace evaluation stack ensuring consistent end-of-episode semantics across (i) dataset serialization, (ii) token-to-symbol mappings, and (iii) logical evaluation, thereby preventing semantic mismatches between model training and temporal-logic assessment.
- An empirical evaluation against diverse temporal specifications in navigation domains, highlighting trade-offs between performance and constraint satisfaction.

2. Background

2.1. Linear Temporal Logic over finite traces

Linear Temporal Logic (LTL) [10] is a formal language that extends traditional propositional logic with modal operators, allowing the specification of rules that must hold *through time*. Here, we use LTL over finite traces (LTL_f) [9], a popular variant of LTL which models finite, but length-unbounded, traces of executions, making it suitable for finite-horizon problems. Given a finite set Σ of atomic propositions, the set of LTL_f formulas φ is inductively defined as follows:

$$\varphi ::= \top \mid \perp \mid a \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid X\varphi \mid \varphi_1 U \varphi_2,$$

where $a \in \Sigma$. We use $\top$ and $\perp$ to denote *True* and *False*, respectively. X (Strong Next) and U (Until) are temporal operators. Other temporal operators are N (Weak Next) and R (Release), defined as $N\varphi \equiv \neg X\neg\varphi$ and $\varphi_1 R \varphi_2 \equiv \neg(\neg\varphi_1 U \neg\varphi_2)$; G (globally) $G\varphi \equiv \perp R \varphi$; and F (eventually) $F\varphi \equiv \top U \varphi$.

A trace $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_T)$ is a sequence of propositional assignments to the propositions in Σ , where $\sigma_t \in 2^\Sigma$ is the set of all and only propositions that are true at instant t . Additionally, $|\sigma|$ denotes the length of the trace. Since every trace is finite, $|\sigma| < \infty$; ϵ denotes the empty trace. Note that, if the propositional symbols in Σ are all *mutually exclusive*, i.e., the domain produces exactly one symbol true at each step, then we have $\sigma_t \in \Sigma$. Let $last = |\sigma| - 1$. We inductively define when an LTL_f formula φ is true at an instant i (for $0 \leq i \leq last$), written $\sigma, i \models \varphi$, as follows:

- $\sigma, i \models a$ iff $a \in \sigma_i$, for $a \in \Sigma$;
- $\sigma, i \models \neg\varphi$ iff $\sigma, i \not\models \varphi$;
- $\sigma, i \models \varphi_1 \wedge \varphi_2$ iff $\sigma, i \models \varphi_1$ and $\sigma, i \models \varphi_2$;

- $\sigma, i \models X\varphi$ iff $i < \text{last}$ and $\sigma, i+1 \models \varphi$;
- $\sigma, i \models \varphi_1 \mathcal{U} \varphi_2$ iff for some j such that $i \leq j \leq \text{last}$, we have $\sigma, j \models \varphi_2$, and for all k such that $i \leq k < j$, we have $\sigma, k \models \varphi_1$.

We say that σ satisfies φ , written $\sigma \models \varphi$, if $\sigma, 0 \models \varphi$. A formula φ is satisfiable if it is true wrt some trace σ , and is valid, if it is true wrt every trace σ .

Any LTL_f formula φ can be translated into a Deterministic Finite Automaton (DFA) [9] $A_\varphi = (2^\Sigma, Q, q_0, \delta, F)$, where 2^Σ is the automaton alphabet, Q is the finite set of states, $q_0 \in Q$ is the initial state, $\delta : Q \times 2^\Sigma \rightarrow Q$ is the transition function, and $F \subseteq Q$ is the set of final states. Additionally, we recursively define the extended transition function over traces $\delta^* : Q \times (2^\Sigma)^* \rightarrow Q$ as:

$$\begin{aligned} \delta^*(q, \epsilon) &= q \\ \delta^*(q, \sigma \cdot \mathbf{x}) &= \delta^*(\delta(q, \sigma), \mathbf{x}), \end{aligned} \quad (1)$$

where $\sigma \in 2^\Sigma$ is a symbol and $\mathbf{x} \in (2^\Sigma)^*$ is a trace. The automaton accepts the trace σ iff $\delta^*(q_0, \sigma) \in F$, and in that case we say that σ belongs to the language of the automaton, denoted as $L(A_\varphi)$. We have that φ and A_φ are equivalent because, for any trace $\sigma \in (2^\Sigma)^*$:

$$\sigma \in L(A_\varphi) \iff \sigma \models \varphi. \quad (2)$$

2.2. Offline Reinforcement Learning

In Reinforcement Learning (RL), the interaction between an agent and the environment is commonly modeled as a Markov Decision Process [1], defined by the tuple (S, A, t, r, γ) where S is the set of states, A the set of actions, $t : S \times A \times S \rightarrow [0, 1]$ the transition function s.t. $\sum_{s' \in S} t(s, a, s') = 1$, $r : S \times A \rightarrow R$ the reward function, and $\gamma \in [0, 1]$ the discount factor. A policy $\pi : S \rightarrow A$ maps states to actions; the value function $V^\pi(s)$ denotes the expected discounted return obtained by following π from s . The goal of the agent is to learn the optimal policy π^* maximizing expected discounted return.

Traditional RL methods assume direct interaction with the environment during training. The agent collects experience by executing actions, observing state transitions and rewards, and updating its policy accordingly. This online interaction paradigm enables iterative improvement but may be costly, unsafe or even unfeasible in real-world applications. This motivates the study of offline RL [3], where the agent is given a fixed dataset $\mathcal{D}$ of trajectories generated by some (unknown) behavior policy, and must learn a policy without additional environment interaction. In order to prevent issues with out-of-distribution data, many offline RL algorithms incorporate mechanisms to constrain learned policies to remain close to the support data distribution [11].

2.3. Reinforcement Learning as Sequence Modeling

Some works [5, 6] propose an alternative perspective on RL, viewing it as a sequence modeling problem rather than a value estimation. In [5] a trajectory is treated as a sequence of tokens and an autoregressive model is used to learn the next token in the sequence. Suppose we have N -dimensional states and M -dimensional actions. A trajectory τ of horizon T can be represented as

$$\tau = (s_1^1, s_1^2, \dots, s_1^N, a_1^1, \dots, a_1^M, r_1, s_2^1, \dots, r_T) \quad (3)$$

Subscripts on all tokens denote timestep and superscripts on states and actions denote dimension (i.e., s_t^i is the i^{th} dimension of the state at time t). An autoregressive model is capable of estimating the probability of the t^{th} token x_t given the set of all past tokens $\mathbf{x}_{<t}$:

$$P(x_t \mid x_1, x_2, \dots, x_{t-1}) = P(x_t \mid \mathbf{x}_{<t}). \quad (4)$$

Instead of learning value functions or policies, one can learn the joint distribution

$$P(\mathbf{x}) = \prod_{i=1}^{|\tau|} P(x_i \mid \mathbf{x}_{<i}). \quad (5)$$

At each generation step, a token must be *sampled* from the next activity probability. A common way of selecting the next token to feed into the autoregressor is to *greedily* choose it by maximizing the next token probability at each step, as follows:

$$x_t = \underset{x}{\operatorname{argmax}} P(x_t \mid \mathbf{x}_{<t}), \quad \forall t < |\tau|. \quad (6)$$

In general, this greedy search strategy is sub-optimal, because it may not produce the trace that maximizes the joint probability distribution in Equation 5. Other sub-optimal sampling strategies commonly used for this task include Beam Search, Random Sampling, and Temperature Sampling [12].

Trajectory Transformer (TT) models are transformer-based autoregressive architectures well suited for addressing RL as sequence modeling. In the offline setting, such models can be trained via maximum likelihood using teacher forcing, by maximizing the log-probability of trajectories in the dataset. At inference time, planning can be performed by sampling or beam search, optionally biasing candidate trajectories according to cumulative reward or reward-to-go estimates. This paradigm unifies dynamics modeling, policy learning, and behavior regularization within a single sequence model. Note that this approach focuses exclusively on maximizing return and modeling the data distribution, but they may violate the constraints that emerge from (or we want to impose to) the domain. This motivates our neurosymbolic framework guiding the learned trajectory distribution toward constraint satisfaction.

3. Related Work

3.1. Offline RL as sequence modeling

Offline Reinforcement Learning [3] is frequently addressed by techniques that are based on value functions [11, 13], goal-conditioned behavior cloning [14, 15] and reward-conditioned behavior cloning [16, 17]. As previously explained, recent works show how this problem can be tackled as a sequence model problem. In [18] RL via supervised learning is analyzed, this work shows that sometimes pure supervised learning is competitive wrt more complex architectures, given careful tuning of the policy capacity. In contrast, other recent works have adopted the use of Transformers [19] as sequence prediction models, and they have provided a promising direction for offline RL, which is also flexible enough to be combined with mentioned techniques.

In particular, Trajectory Transformers (TTs) [5] and Decision Transformers (DTs) [6] follow this idea by using beam-search planning and reward conditioning, respectively. Notably, these architectures are further studied combining techniques based on Dynamic Programming (such as Q-learning) [20], online fine-tuning [21], few-shot policy generalization [22], and automatically-generated waypoints [23]. There are also works that do not build directly on TTs or DTs, such as [24], where Transformers are used to overcome inefficiencies of Diffusion Models.

3.2. Safe Reinforcement Learning

Safe RL focuses on methods for ensuring that learned policies satisfy some given safety constraints. Online RL techniques [25, 26] typically model safety requirements as constraints on the expected cost, and enforce safety through constrained optimization or Lagrangian formulations during policy learning. Recently, also offline Safe RL is gaining popularity, including techniques for both soft constraints [27] and hard constraints [28]. Adaptations of DTs and TTs are also studied in the context of offline Safe RL [29, 30, 31, 32], confirming the growing interest in these models in recent years. These works share a common formulation grounded in Constrained MDPs, where safety is enforced by conditioning the policy on cost signals and constraint thresholds. Despite their differences, all these approaches operate by modifying the conditioning signals or relabeling trajectory returns to steer generation toward safe behavior. In contrast, our method leaves the trajectory data and its reward/cost structure untouched and instead regularizes the training objective with a differentiable logic loss derived from DFA progression, injecting temporal-logic knowledge directly into the learning process rather than into

the data representation. Recent work [33] also addresses RL using TTs and LTL_f constraints, but their approach differs from ours as it relies on reward shaping derived from DFA states to imitate trajectories.

An alternative line of work enforces safety through shielding [34, 35, 36, 37], where a runtime monitor prevents the agent from executing actions that may lead to unsafe states. While shielding provides strong safety guarantees, it typically assumes the availability of an explicit environment model, it is primarily designed for on-policy learning, and it is structurally limited to the "safety fragment" of formal specifications. In contrast, our method is well suited for offline RL, does only require knowledge of the constraints without assuming any other knowledge of the environment, and can be used to impose any LTL_f formula as a constraint, also the ones outside the safety fragment.

3.3. Temporally-constrained sequence generation

Recently, there has been increasing interest in constraining autoregressive sequence generation through logical knowledge. Applications span Cyber-Physical Systems [38], Business Process Management [7, 39], and natural language generation with Large Language Models (LLMs) [40, 41, 42, 43, 44, 45]. Most prior work incorporates constraints at *test time*, guiding suffix generation via constrained beam search [39, 40, 41, 42], auxiliary tractable models [46, 47], or conditioned sampling [43, 44]. In contrast, we enforce constraints at *training time* through an auxiliary logical loss. Only few approaches follow this direction [38, 7, 45, 48]. Among them, [7, 45] estimate the probability of satisfying a formula via pseudoprobabilities or Monte Carlo; STLnet [38] relies on a student-teacher scheme that is difficult to apply in discrete domains [7]; finally, [48] impose a differentiable loss computed from LTL specifications.

A key limitation of existing methods is that they target *homogeneous* sequences (e.g., text), assuming that logical formulas are directly expressed over the generated tokens. While reasonable for language, this assumption is limiting in RL settings. Here, we consider trajectories of states, actions, and rewards (as formalized in Eq. 3), where constraints are naturally specified over abstract, high-level *symbols* that must be grounded in subsequences representing states or state-action pairs. Our approach builds on [7, 8], which approximate LTL_f satisfaction via Monte Carlo and use it as a training loss.

4. Method

4.1. Problem Formulation

In offline RL, the agent is provided with a fixed dataset $\mathcal{D} = \{\tau^{(i)}\}_{i=1}^N$ of previously collected trajectories. Each trajectory is a finite sequence of transitions and an *end-of-trace* symbol EOT^1 , i.e. $\tau = (y^1, \dots, y^T, EOT)$, where each transition token y^j group encodes state, action, reward, and optional environment-specific auxiliary fields (e.g., safety cost). Here, we also consider some prior knowledge about the task, expressed as LTL_f constraints, to capture high-level behavioral requirements. Given an LTL_f formula φ , our objective is to learn a trajectory generation policy that maximizes both the expected return of the policy and the probability that generated trajectories satisfy φ . We define atomic propositions through an environment-specific extractor

$$\Pi(y^t) \in \{0, 1\}^K$$

and φ is defined over these K propositional symbols. In our experiments proposition extraction is *state-centric*, i.e. φ is defined only over state dimensions, but our approach supports also imposing constraints on action, state-action pairs and auxiliary-terms.

A logic adapter is used to define the authoritative token schema and the unique end token identifier shared by dataset builders, token-to-symbol mapping, DFA rollout, and logic-loss evaluation. Satisfaction evaluation follows the standard LTL_f semantics (see Sec. 2.1) and is therefore defined on complete finite traces. This design avoids inconsistent termination handling across modules and ensures agreement

¹ LTL_f is interpreted over finite traces. We append EOT to each trajectory to explicitly mark its termination. The DFA consumes the trajectory symbols up to EOT , and satisfaction is evaluated by whether the automaton reaches an accepting state.

between hard DFA evaluation and differentiable soft evaluation. As explained in the background, we assume an autoregressive neural model f_θ with trainable parameters θ that estimates an approximation P_θ of the probability of the next event τ_t given a trace of previous events $\tau_{<t}$ (Equation 4):

$$\begin{aligned}\tilde{y}_t &= f_\theta(\tau_{<t}) \\ P(\tau_t = v_i \mid \tau_{<t}) &\approx \tilde{y}_t[i].\end{aligned}\tag{7}$$

where, τ_t denotes the token at position t , and v_i is an element of the token vocabulary. Note that we do not make any assumptions about the neural model. The model parameters are typically trained using a supervised loss $L_{\mathcal{D}}$, evaluated on a dataset $\mathcal{D}$ of ground-truth traces obtained by observing the process. The loss for a trace $\tau \in \mathcal{D}$ of length T is defined as follows:

$$L_{\mathcal{D}}(\tau) = \frac{1}{T} \sum_{t=1}^T \text{cross-entropy}(f_\theta(\tau_{<t}), \tau_t).\tag{8}$$

The goal is to maximize the probability $P_{\theta \models \varphi}$ that sampled traces $\tau \sim P_\theta$ satisfy the specification:

$$P_{\theta \models \varphi} = \mathbb{E}_{\tau \sim P_\theta}[\tau \models \varphi] = \sum_{\tau} P_\theta(\tau) \mathbb{I}\{\tau \models \varphi\}.\tag{9}$$

Computing this probability exactly is infeasible, since it would require to enumerate all possible traces of maximum length T . In [7, 8] a differentiable procedure to approximate the previous probability at each optimization step of the autoregressor is designed. They introduce a logic loss function L_φ , and compute the training objective as a linear combination of the logic and the supervised loss:

$$L = \alpha L_\varphi + (1 - \alpha) L_{\mathcal{D}},\tag{10}$$

with α being a trade-off parameter between 0 and 1 that balances the influence of each loss on the training process. Here we adapt the logic loss computation to offline RL domains and apply it at training time, together with other techniques at test time to maximize adherence with the constraints.

4.2. Automata-Based Satisfaction and Logic Loss

Now, we can discuss how to approximate the target probability in Eq. 9. Specifically, [8] uses a Monte Carlo estimation by sampling a set of complete traces $\{\tau^{(1)}, \tau^{(2)}, \dots, \tau^{(N)}\} \sim P_\theta$ according to the distribution learned by the autoregressor, and compute an approximation of the probability as follows:

$$\hat{P}_{\theta \models \varphi} = \frac{1}{N} \sum_{i=1}^N \mathbb{I}\{\tau^{(i)} \models \varphi\}.\tag{11}$$

Note that the sampled traces $\tau^{(i)}$ are generated by a NN that returns a probability distribution at each time step, and thus they are replaced by a probabilistic counterpart $\tilde{\tau}^{(i)}$, where symbols are sampled from that probability distribution. Moreover, the indicator function $\mathbb{I}$ in Equation 11 is replaced by a method that computes the (probabilistic) compliance of a probabilistic trace with the knowledge. To achieve this, we can leverage on (i) the Gumbel-Softmax sampling, that generates differentiable, near one-hot suffixes during training and (ii) DeepDFA [49], that encodes temporal logic properties as a recurrent layer, enabling efficient and differentiable evaluation of logical constraints.

Gumbel-Softmax sampling. To sample from the probability distribution returned by a NN, but still maintain differentiability, we can use the Gumbel-Softmax sampling, which produces approximations of discrete symbols as one-hot-like vectors. Given the predicted token distribution $\tilde{y}_t$, we get

$$\tilde{x}_t = \text{softmax}\left(\frac{\log(\tilde{y}_t) + G}{temp}\right)\tag{12}$$

where G is a Gumbel noise vector and $temp$ is a temperature parameter controlling the sharpness of the distribution. These soft tokens are then fed into the differentiable DFA, which computes the probability that the sampled trajectory satisfies the LTLf specification. As $temp \rightarrow 0$, the output approaches a discrete one-hot vector, while for $temp = 1$, it remains close to the original continuous probabilities in $\tilde{y}_t$. Since the next activity is only *probabilistically* grounded, we denote it as $\tilde{\tau}_t$.

DeepDFA. As explained in Sec. 2.1, each LTL_f constraint φ can be compiled into an equivalent DFA A_φ . We extend the DFA to handle the special EOT symbol, such that only traces including EOT are accepted. There exists automatic translation tools, such as `lTL2DFA` [50], that can be used off-the-shelf. Many alternatives exist in the literature to evaluate continuous satisfaction of temporal constraints over sequences of *soft* (probabilistic or fuzzy) symbol assignments [51, 49, 52, 53]. In our work we used DeepDFA [49], following what has been done in [7, 8]. DeepDFA is a neural, probabilistic relaxation of a standard DFA, where the automaton is represented in matrix form and the input symbols, states, and outputs are *probabilistically grounded*. This allows us to compute the probabilistic compliance $P_{\text{DDFA}}(\tilde{\tau}^{(i)} \models \varphi)$ of a sampled trace $\tilde{\tau}^{(i)}$ with the knowledge φ . Therefore, Equation 11 becomes:

$$\hat{P}_{\theta \models \varphi} = \frac{1}{N} \sum_{i=1}^N P_{\text{DDFA}}(\tilde{\tau}^{(i)} \models \varphi). \quad (13)$$

Finally, we compute the logic loss, which enforces the satisfaction of prior knowledge over entire traces:

$$L_\varphi = -\log \left(\hat{P}_{\theta \models \varphi} \right). \quad (14)$$

4.3. Instantiating the Framework for TT and DT

The mechanism above requires a differentiable trace of soft symbols to roll out the DFA. How this trace is obtained depends on what the underlying autoregressive model generates. TTs predict the full trajectory token-by-token, including state dimensions. DTs, by contrast, are conditioned on returns, states, and past actions, and predict only the action distribution; the state at the next timestep is not generated by the model. To recover the state evolution needed for the DFA rollout, we pair the DT with a separately learned, differentiable model of the environment dynamics. This preserves an end-to-end differentiable pipeline while requiring, for DT-based instantiations, an auxiliary dynamics model.

5. Experimental Evaluation

To evaluate the framework, we conduct experiments² on a controlled navigation benchmark designed to simulate safety-critical decision making under temporally extended constraints. The experiments focus on how the logic regularization influences the behavior of transformer-based policies and how different levels of logical guidance affect the trade-off between task performance and constraint satisfaction.

Environment

We evaluate our approach on the ColourBomb environment [4], a grid-based navigation domain illustrated in Figure 1. The environment contains:

- **Start cell:** the initial location of the agent.
- **Goal cells:** terminal states (with different colours) providing positive rewards.
- **Bomb cells:** hazardous states that terminate the episode with negative reward.
- **Walls:** non-traversable cells that block movement.

The agent must navigate from the start cell to one of the goal cells while avoiding bombs and walls. The environment is episodic with a finite horizon. Each step incurs a small penalty to encourage efficient navigation, while reaching a goal yields positive reward and entering a bomb cell yields negative reward.

²The code is publicly available at: https://github.com/ashkanans/nesy_rl_test.

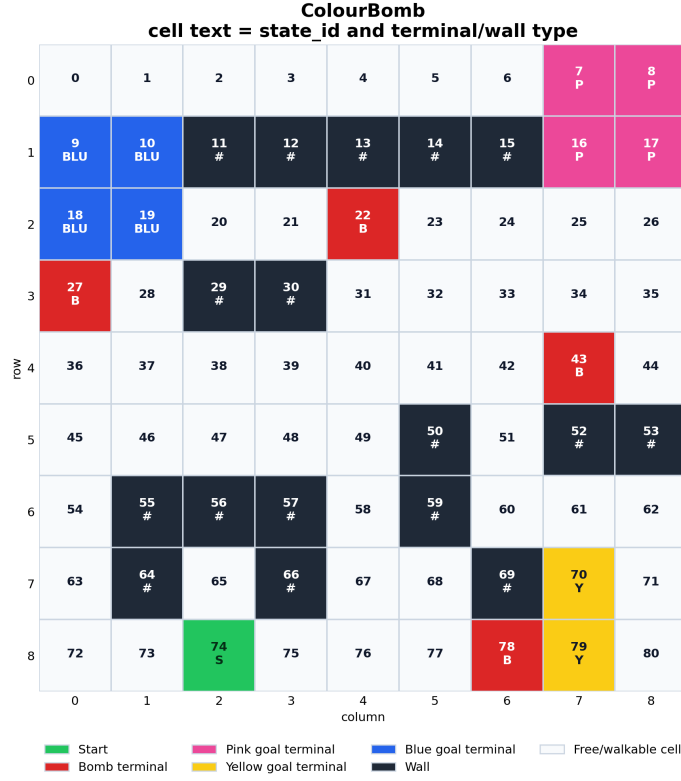


Figure 1: ColourBomb environment structure.

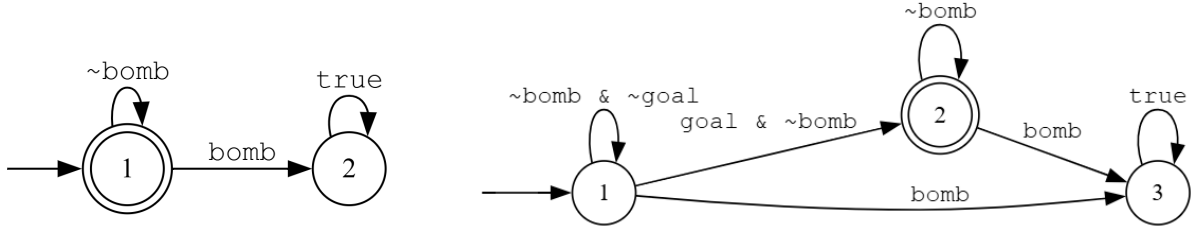


Figure 2: DFAs for Safety and Reach-while-Safe constraints. The incoming arrow denotes the initial state; double circles denote accepting states. Edges are labeled by propositions/symbols that trigger transitions. A trace satisfies the formula if, after consuming all symbols up to EOT, the DFA is in an accepting state.

Temporal Specifications

We evaluate the system using two representative families of LTL_f specifications:

- **Safety:** requires the agent to avoid bomb cells throughout the entire trajectory, i.e. $G(\neg bomb)$
- **Reach-while-Safe:** requires the agent to eventually reach a goal while always avoiding bomb, i.e. $G(\neg bomb) \wedge F(goal)$

Note that $F(goal)$ is a liveness property in temporal logics terminology. Note also that, since we assume (i) goal is terminal, (ii) goal is not a bomb, and (iii) traces stop when goal is reached, then we could rewrite the constraint as $(\neg bomb) U goal$, which ensures that the agent remains in safe states until a goal is reached. We show the automata for to these two specifications in Fig. 2.

These specifications allow us to analyze the behavior of the method under both purely safety-oriented and mixed safety-and-performance objectives. One could also consider more complex formulas, for instance requiring the agent to visit coloured states in a given order. Notably, our framework goes beyond most papers on Safe RL, allowing the specification of arbitrary temporal constraints. In future work, we aim to explore the use of combined formulas drawn from *Declare* patterns [54].

Methods Compared

We evaluate variants of the TT and DT architectures with different levels of logical regularization. We consider vanilla TT and DT as the baseline model, without any logical regularization. This corresponds to setting $\alpha = 0$ to Eq. 10. Then, we impose logic regularization, that is, the training objective combines the standard sequence-prediction loss with the proposed logic loss computed by the use of differentiable DFAs. To control the weight of logic regularization and study how it affects return and constraint satisfaction, we vary the trade-off parameter α . All models use greedy decoding at inference time to isolate the effect of the logic regularization during training.

Evaluation Metrics

We evaluate policies using several metrics capturing both task performance and logical compliance:

- **Return:** the average episodic reward obtained by the agent.
- **Satisfaction Rate:** the fraction of trajectories that satisfy the LTL_f specification.
- **Goal Rate:** the proportion of episodes in which the agent successfully reaches a goal state.
- **Bomb Hit Rate:** the fraction of trajectories that terminate by entering a bomb cell.

Experiments on Decision Transformers

Here, we report the preliminary results obtained by DTs. The DT product-value experiments are aggregated over three independent random seeds.

Invariant Safety. We report Decision Transformer (DT) alpha-sweep results with greedy decoding. For the invariant safety specification $G(\neg bomb)$, vanilla DT consistently reaches bomb states and obtains zero satisfaction. All logic-regularized settings remove bomb hits and satisfy the safety formula. Using satisfaction as the primary criterion and return as a tie-breaker, the best configuration is $\alpha = 0.4$.

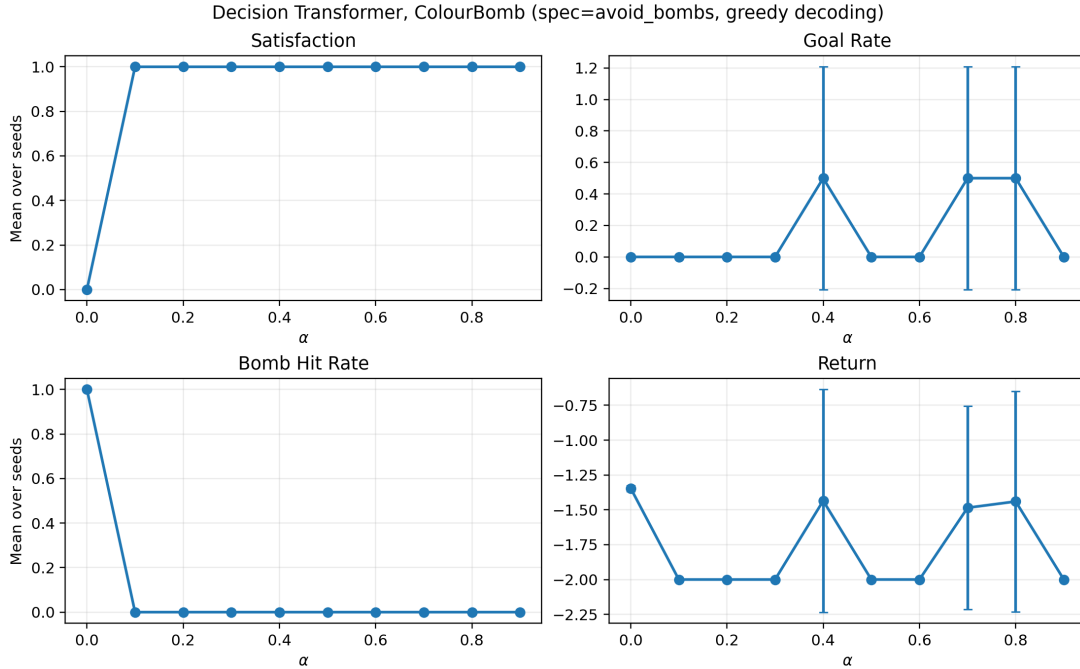


Figure 3: DT results for the invariant safety constraint as a function of α .

Reach-while-Safe. For the conjunctive specification $G(\neg bomb) \wedge F(goal)$, the setting is harder because safety and goal achievement must hold together. Vanilla DT again fails the specification by reaching bomb states and never reaching a goal. Intermediate values of α show unstable behavior across seeds: some configurations avoid bombs but time out, while others reach the goal safely. For $\alpha \geq 0.6$, the model satisfies the full reach-while-safe specification in both seeds. The best configuration under the same satisfaction-then-return criterion is $\alpha = 0.9$.

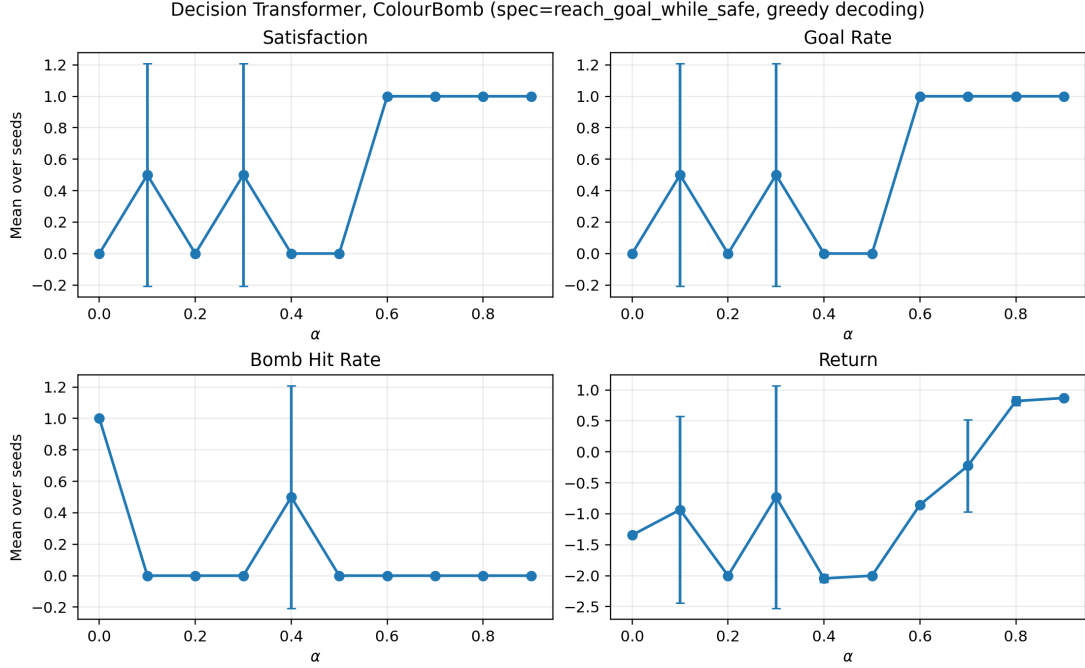


Figure 4: DT results for the Reach-while-Safe constraint as a function of α .

Quantitative Summary. Table 1 summarizes the vanilla baseline and the best logic-regularized configuration for each specification. Logic regularization eliminates bomb hits for both objectives. For the safety-only formula, the selected configuration satisfies the safety constraint but reaches the goal in only one of the seeds. For the reach-while-safe formula, $\alpha = 0.9$ achieves perfect satisfaction, zero bomb-hit rate, and consistent goal reaching, substantially improving return over vanilla DT.

Spec	Setting	Satisfaction	Goal	Bomb Hit	Return
safety	vanilla ($\alpha = 0$)	0.000	0.000	1.000	-1.345
safety	logic ($\alpha = 0.4$)	1.000	0.500	0.000	-1.435
reach-while-safe	vanilla ($\alpha = 0$)	0.000	0.000	1.000	-1.345
reach-while-safe	logic ($\alpha = 0.9$)	1.000	1.000	0.000	0.870

Table 1

Decision Transformer key results.

Experiments on Trajectory Transformers

We now present preliminary results obtained by our implementation based on Trajectory Transformers.

Invariant Safety. We first analyze the invariant safety specification $G(\neg bomb)$. Figure 5 shows that small values of α do not improve safety compared to the baseline, while larger values provide measurable benefits. Setting $\alpha = 0.8$ achieves the highest satisfaction rate. In general, the logic-regularized policy

reduces the probability of entering bomb cells and improves overall compliance with the specification. At the same time, it does not affect task performance, for which our method maintains comparable results. Notably, setting $\alpha = 0.8$ improves also the return wrt the baseline.

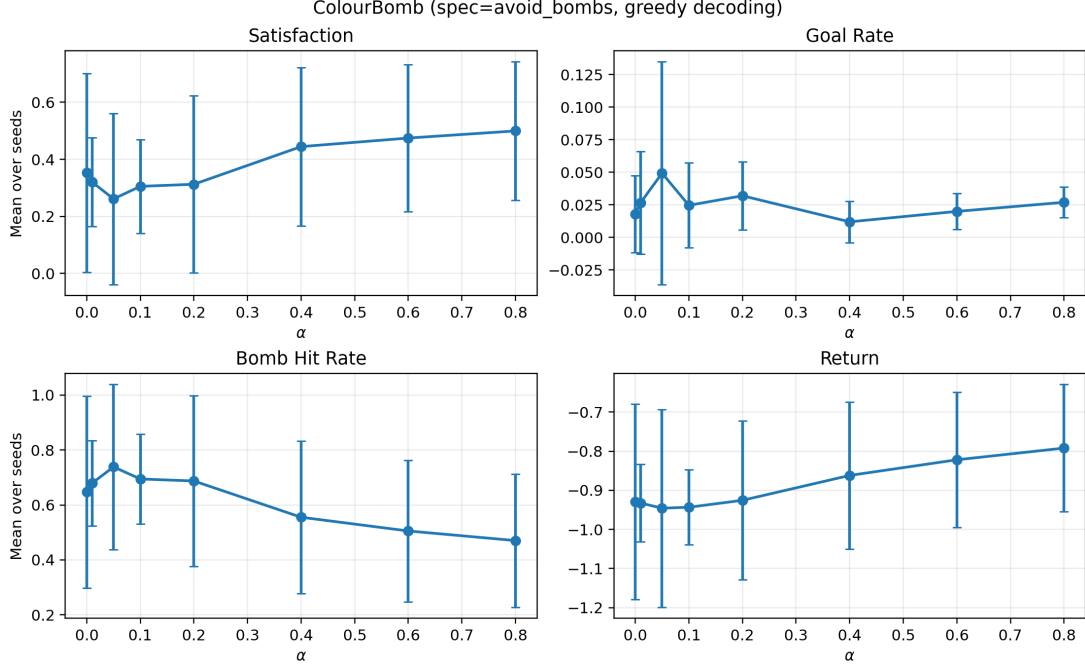


Figure 5: Results under the safety constraint as a function of the weighting loss parameter α .

Reach-while-Safe. We next include the goal state in the LTL_f constraint by the specification $G(\neg bomb) \wedge F(goal)$, which requires the agent to reach a goal while remaining safe throughout the trajectory. Figure 6 summarizes the results for this specification. As expected, satisfying the constraint becomes significantly harder. As a result, the satisfaction rate is substantially lower than in the invariant safety case. Nevertheless, we can still observe that our approach outperforms the baseline for all values of $\alpha \geq 0.1$, with $\alpha = 0.1$ providing the best results. This shows that logic regularization still leads to improvements for more complex specifications. It is worth highlighting that, also in this case, the return associated with values $\alpha \geq 0.1$ is slightly better than what we observe for the baseline.

Cross-Specification Comparison. Table 2 compares the best configurations for each specification. The invariant safety objective can be improved primarily by reducing bomb-hit events, which directly increases the probability of satisfying the safety constraint. In contrast, the Reach-while-Safe objective remains limited by the difficulty of achieving the goal while maintaining safety simultaneously. In many trajectories, the agent avoids bombs successfully but fails to reach the goal within the episode horizon. This result highlights the intrinsic trade-off between safety and task completion in environments where safe paths towards the goal may be longer or harder to discover.

Quantitative Summary. Table 2 reports representative configurations illustrating the trade-offs observed in the experiments. For the invariant safety task, the logic-regularized model with $\alpha = 0.8$ increases both the satisfaction rate, the goal rate, and the return. For the Reach-while-Safe specification, the best configuration is $\alpha = 0.1$, where our approach again improves both the satisfaction rate and the return. It is worth noting that including the reachability of the goal state in the trace constraint helps us in improving the goal-reached rate itself, while slightly degrading performances with respect to the bomb-hit rate. These results confirm that incorporating logical knowledge into the training objective can improve safety-related metrics while preserving competitive performance in both DT and

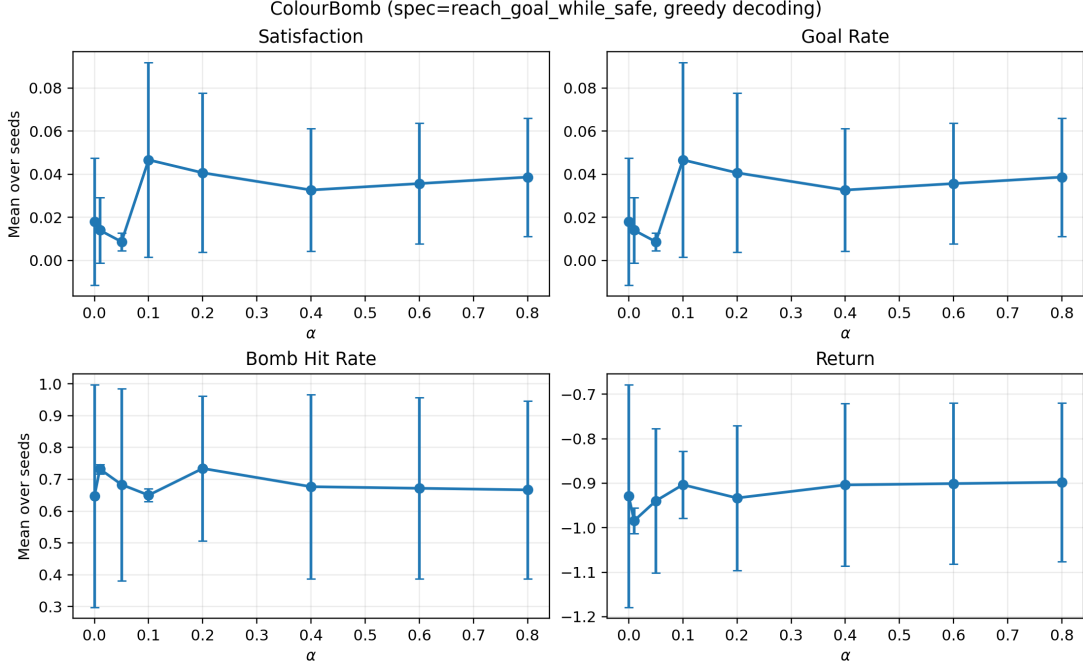


Figure 6: Results under the Reach-while-Safe constraint as a function of the weighting loss parameter α .

Spec	Setting	Satisfaction	Goal	Bomb Hit	Return
safety	vanilla ($\alpha = 0$)	0.353	0.018	0.647	-0.929
safety	logic ($\alpha = 0.8$)	0.499	0.027	0.470	-0.792
reach-while-safe	vanilla ($\alpha = 0$)	0.018	0.018	0.647	-0.929
reach-while-safe	logic ($\alpha = 0.1$)	0.047	0.047	0.650	-0.903

Table 2

Trajectory Transformer key results.

TT. In both cases, tuning the α parameter remains crucial for achieving high performance. However, compared to DT, the improvements observed in the TT experiments are surprisingly more moderate. We leave to future work a deeper investigation into the reasons behind this difference.

6. Conclusion

We presented a neurosymbolic framework for injecting LTL_f constraints into transformer-based policies for offline RL. Our approach aims at constructing an automaton-aware decision mechanisms that guide trajectory generation, and it relies on two features: a differentiable representation of DFAs and a differentiable logic loss that regularizes the training objective. We applied the proposed framework to Trajectory Transformers and Decision Transformers. Unlike related popular approaches, we don't need to engineer the underlying reward function, nor we need to focus on a small subset of temporal specifications. Preliminary experiments on the ColourBomb navigation benchmark show that logic regularization can improve constraint satisfaction while maintaining competitive task performance. These findings suggest that integrating symbolic background knowledge into offline sequence-based RL is a promising direction for improving policy reliability in safety-critical domains. Future work will extend the empirical evaluation to additional benchmarks, model architectures, temporal specifications, and decoding mechanisms. It would be significant also to perform an extensive end-to-end comparison with well-established frameworks for Safe RL.

Acknowledgments

This work has been supported by the the PNRR MUR project FAIR (No. PE0000013), and the Italian National Ph.D. on Artificial Intelligence at Sapienza University of Rome.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT and Claude in order to: Improve writing style. After using these services, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

- [1] R. S. Sutton, A. G. Barto, et al., Reinforcement learning: An introduction, volume 1, MIT press Cambridge, 1998.
- [2] A. Kushwaha, K. Ravish, P. Lamba, P. Kumar, A survey of safe reinforcement learning and constrained mdps: A technical survey on single-agent and multi-agent safety, CoRR abs/2505.17342 (2025). URL: <https://doi.org/10.48550/arXiv.2505.17342>. doi:10.48550/ARXIV.2505.17342. arXiv:2505.17342.
- [3] R. F. Prudencio, M. R. O. A. Máximo, E. L. Colombini, A survey on offline reinforcement learning: Taxonomy, review, and open problems, IEEE Trans. Neural Networks Learn. Syst. 35 (2024) 10237–10257. URL: <https://doi.org/10.1109/TNNLS.2023.3250269>. doi:10.1109/TNNLS.2023.3250269.
- [4] E. H.-D. Le Court, F. Belardinelli, A. W. Goodall, Probabilistic shielding for safe reinforcement learning, in: Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI Press, 2025.
- [5] M. Janner, Q. Li, S. Levine, Offline reinforcement learning as one big sequence modeling problem, in: M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, J. W. Vaughan (Eds.), Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, 2021, pp. 1273–1286.
- [6] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, I. Mordatch, Decision transformer: reinforcement learning via sequence modeling, in: Proceedings of the 35th International Conference on Neural Information Processing Systems, NIPS '21, Curran Associates Inc., Red Hook, NY, USA, 2021.
- [7] E. Umili, G. P. Licks, F. Patrizi, Enhancing deep sequence generation with logical temporal knowledge, in: Proceedings of the 3rd International Workshop on Process Management in the AI Era (PMAI 2024) co-located with 27th European Conference on Artificial Intelligence (ECAI 2024), Santiago de Compostela, Spain, October 19, 2024, 2024, pp. 23–34. URL: <https://ceur-ws.org/Vol-3779/paper4.pdf>.
- [8] A. Mezini, E. Umili, I. Donadello, F. M. Maggi, M. Mancanelli, F. Patrizi, Neuro-symbolic predictive process monitoring, arXiv preprint arXiv:2509.00834 (2025).
- [9] G. De Giacomo, M. Y. Vardi, Linear temporal logic and linear dynamic logic on finite traces, in: F. Rossi (Ed.), IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013, IJCAI/AAAI, 2013, pp. 854–860. URL: <http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6997>.
- [10] A. Pnueli, The temporal logic of programs, in: 18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977, IEEE Computer Society, 1977, pp. 46–57. URL: <https://doi.org/10.1109/SFCS.1977.32>. doi:10.1109/SFCS.1977.32.
- [11] A. Kumar, A. Zhou, G. Tucker, S. Levine, Conservative q-learning for offline reinforcement learning, Advances in neural information processing systems 33 (2020) 1179–1191.

- [12] E. Rama-Maneiro, F. Patrizi, J. C. Vidal, M. Lama, Towards learning the optimal sampling strategy for suffix prediction in predictive monitoring, in: G. Guizzardi, F. M. Santoro, H. Mouratidis, P. Soffer (Eds.), *Advanced Information Systems Engineering - 36th International Conference, CAiSE 2024, Limassol, Cyprus, June 3-7, 2024, Proceedings*, volume 14663 of *Lecture Notes in Computer Science*, Springer, 2024, pp. 215–230.
- [13] I. Kostrikov, A. Nair, S. Levine, Offline reinforcement learning with implicit q-learning, *arXiv preprint arXiv:2110.06169* (2021).
- [14] A. V. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, S. Levine, Visual reinforcement learning with imagined goals, *Advances in neural information processing systems* 31 (2018).
- [15] Y. Ding, C. Florensa, P. Abbeel, M. Phielipp, Goal-conditioned imitation learning, *Advances in neural information processing systems* 32 (2019).
- [16] A. Kumar, X. B. Peng, S. Levine, Reward-conditioned policies, *arXiv preprint arXiv:1912.13465* (2019).
- [17] R. K. Srivastava, P. Shyam, F. Mutz, W. Jaśkowski, J. Schmidhuber, Training agents using upside-down reinforcement learning, *arXiv preprint arXiv:1912.02877* (2019).
- [18] S. Emmons, B. Eysenbach, I. Kostrikov, S. Levine, Rvs: What is essential for offline rl via supervised learning?, *arXiv preprint arXiv:2112.10751* (2021).
- [19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, 2017*, pp. 5998–6008.
- [20] T. Yamagata, A. Khalil, R. Santos-Rodriguez, Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline rl, in: *International Conference on Machine Learning*, PMLR, 2023, pp. 38989–39007.
- [21] Q. Zheng, A. Zhang, A. Grover, Online decision transformer, in: *international conference on machine learning*, PMLR, 2022, pp. 27042–27059.
- [22] M. Xu, Y. Shen, S. Zhang, Y. Lu, D. Zhao, J. Tenenbaum, C. Gan, Prompting decision transformer for few-shot policy generalization, in: *international conference on machine learning*, PMLR, 2022, pp. 24631–24645.
- [23] A. Badrinath, Y. Flet-Berliac, A. Nie, E. Brunskill, Waypoint transformer: Reinforcement learning via supervised learning with intermediate targets, *Advances in Neural Information Processing Systems* 36 (2023) 78006–78027.
- [24] R. Huang, Y. Pei, G. Wang, Y. Zhang, Y. Yang, P. Wang, H. Shen, Diffusion models as optimizers for efficient planning in offline rl, in: *European Conference on Computer Vision*, Springer, 2024, pp. 1–17.
- [25] J. Achiam, D. Held, A. Tamar, P. Abbeel, Constrained policy optimization, in: *International conference on machine learning*, Pmlr, 2017, pp. 22–31.
- [26] C. Tessler, D. J. Mankowitz, S. Mannor, Reward constrained policy optimization, *arXiv preprint arXiv:1805.11074* (2018).
- [27] H. Xu, X. Zhan, X. Zhu, Constraints penalized q-learning for safe offline reinforcement learning, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 2022, pp. 8753–8760.
- [28] Y. Zheng, J. Li, D. Yu, Y. Yang, S. E. Li, X. Zhan, J. Liu, Safe offline reinforcement learning with feasibility-guided diffusion model, *arXiv preprint arXiv:2401.10700* (2024).
- [29] Z. Liu, Z. Guo, Y. Yao, Z. Cen, W. Yu, T. Zhang, D. Zhao, Constrained decision transformer for offline safe reinforcement learning, in: *International conference on machine learning*, PMLR, 2023, pp. 21611–21630.
- [30] R. Wang, D. Zhou, Safe decision transformer with learning-based constraints, in: *Neurips Safe Generative AI Workshop 2024*, 2024.
- [31] Z. Guo, W. Zhou, W. Li, Temporal logic specification-conditioned decision transformer for offline safe reinforcement learning, *arXiv preprint arXiv:2402.17217* (2024).
- [32] Q. Zhang, L. Zhang, H. Xu, L. Shen, B. Wang, Y. Chang, X. Wang, B. Yuan, D. Tao, Saformer: A

conditional sequence modeling approach to offline safe reinforcement learning, arXiv preprint arXiv:2301.12203 (2023).

- [33] D. Tian, H. Fang, Q. Yang, H. Yu, W. Liang, Y. Wu, Reinforcement learning under temporal logic constraints as a sequence modeling problem, *Robotics and Autonomous Systems* 161 (2023) 104351.
- [34] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, U. Topcu, Safe reinforcement learning via shielding, in: *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.
- [35] N. Jansen, B. Könighofer, S. Junges, A. C. Serban, R. Bloem, Safe reinforcement learning via probabilistic shields, arXiv preprint arXiv:1807.06096 (2018).
- [36] W.-C. Yang, G. Marra, G. Rens, L. De Raedt, Safe reinforcement learning via probabilistic logic shields, arXiv preprint arXiv:2303.03226 (2023).
- [37] F. Belardinelli, A. W. Goodall, et al., Probabilistic shielding for safe reinforcement learning, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 2025, pp. 16091–16099.
- [38] M. Ma, J. Gao, L. Feng, J. A. Stankovic, STLnet: Signal temporal logic enforced multivariate recurrent neural networks, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL: <https://proceedings.neurips.cc/paper/2020/hash/a7da6ba0505a41b98bd85907244c4c30-Abstract.html>.
- [39] C. D. Francescomarino, C. Ghidini, F. M. Maggi, G. Petrucci, A. Yeshchenko, An eye into the future: Leveraging a-priori knowledge in predictive business process monitoring, in: J. Carmona, G. Engels, A. Kumar (Eds.), *Business Process Management - 15th International Conference, BPM 2017, Barcelona, Spain, September 10-15, 2017, Proceedings*, volume 10445 of *Lecture Notes in Computer Science*, Springer, 2017, pp. 252–268. URL: https://doi.org/10.1007/978-3-319-65000-5_15. doi:10.1007/978-3-319-65000-5_15.
- [40] V. Collura, K. Tit, L. Bussi, E. Giunchiglia, M. Cordy, TRIDENT: temporally restricted inference via dfa-enhanced neural traversal, *CoRR* abs/2506.09701 (2025). URL: <https://doi.org/10.48550/arXiv.2506.09701>. doi:10.48550/ARXIV.2506.09701. arXiv:2506.09701.
- [41] X. Lu, P. West, R. Zellers, R. Le Bras, C. Bhagavatula, Y. Choi, NeuroLogic decoding: (un)supervised neural text generation with predicate logic constraints, in: K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, Y. Zhou (Eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Online*, 2021, pp. 4288–4299. URL: <https://aclanthology.org/2021.naacl-main.339/>. doi:10.18653/v1/2021.naacl-main.339.
- [42] X. Lu, S. Welleck, P. West, L. Jiang, J. Kasai, D. Khashabi, R. Le Bras, L. Qin, Y. Yu, R. Zellers, N. A. Smith, Y. Choi, NeuroLogic A*esque decoding: Constrained text generation with lookahead heuristics, in: M. Carpuat, M.-C. de Marneffe, I. V. Meza Ruiz (Eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Seattle, United States*, 2022, pp. 780–799. URL: <https://aclanthology.org/2022.naacl-main.57/>. doi:10.18653/v1/2022.naacl-main.57.
- [43] N. Miao, H. Zhou, L. Mou, R. Yan, L. Li, CGMH: constrained sentence generation by metropolis-hastings sampling, in: *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, AAAI’19/IAAI’19/EAAI’19*, AAAI Press, 2019. URL: <https://doi.org/10.1609/aaai.v33i01.33016834>. doi:10.1609/aaai.v33i01.33016834.
- [44] J. Loula, B. LeBrun, L. Du, B. Lipkin, C. Pasti, G. Grand, T. Liu, Y. Emara, M. Freedman, J. Eisner, R. Cotterell, V. Mansinghka, A. K. Lew, T. Vieira, T. J. O’Donnell, Syntactic and semantic control of large language models via sequential monte carlo, in: *The Thirteenth International Conference on Learning Representations*, 2025. URL: <https://openreview.net/forum?id=xoXn62FzD0>.
- [45] K. Ahmed, K.-W. Chang, G. V. den Broeck, A pseudo-semantic loss for autoregressive models with logical constraints, in: *Thirty-seventh Conference on Neural Information Processing Systems*,

2023. URL: <https://openreview.net/forum?id=hVAla2O73O>.
- [46] B. Krause, A. D. Gotmare, B. McCann, N. S. Keskar, S. Joty, R. Socher, N. F. Rajani, GeDi: Generative discriminator guided sequence generation, in: M.-F. Moens, X. Huang, L. Specia, S. W.-t. Yih (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2021, Association for Computational Linguistics, Punta Cana, Dominican Republic, 2021, pp. 4929–4952. URL: <https://aclanthology.org/2021.findings-emnlp.424/>. doi:10.18653/v1/2021.findings-emnlp.424.
 - [47] H. Zhang, P. Kung, M. Yoshida, G. V. den Broeck, N. Peng, Adaptable logical control for large language models, in: A. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, C. Zhang (Eds.), Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024, 2024. URL: http://papers.nips.cc/paper_files/paper/2024/hash/d15c16cf5619a2b1606da5fc88e3f1a9-Abstract-Conference.html.
 - [48] C. Innes, S. Ramamoorthy, Elaborating on learned demonstrations with temporal logic specifications, arXiv preprint arXiv:2002.00784 (2020).
 - [49] E. Umili, R. Capobianco, Deepdfa: Automata learning through neural probabilistic relaxations, in: ECAI 2024 - 27th European Conference on Artificial Intelligence, 19-24 October 2024, Santiago de Compostela, Spain - Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS 2024), 2024, pp. 1051–1058. URL: <https://doi.org/10.3233/FAIA240596>. doi:10.3233/FAIA240596.
 - [50] F. Fuggitti, LTLf2DFA, 2019. URL: <https://doi.org/10.5281/zenodo.3888410>. doi:10.5281/zenodo.3888410.
 - [51] E. Umili, R. Capobianco, G. De Giacomo, Grounding ltlf specifications in image sequences, in: Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023, Rhodes, Greece, September 2-8, 2023, 2023, pp. 668–678. URL: <https://doi.org/10.24963/kr.2023/65>. doi:10.24963/KR.2023/65.
 - [52] I. Donadello, P. Felli, C. Innes, F. M. Maggi, M. Montali, LTL-based conformance checking of fuzzy event logs, Process Science 2 (2025). doi:10.1007/S44311-025-00020-W.
 - [53] N. Manginas, G. Paliouras, L. D. Raedt, Nesya: Neurosymbolic automata, in: Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2025, Montreal, Canada, August 16-22, 2025, ijcai.org, 2025, pp. 5950–5958. doi:10.24963/IJCAI.2025/662.
 - [54] M. Pesic, W. M. P. van der Aalst, A declarative approach for flexible business processes management, in: J. Eder, S. Dustdar (Eds.), Business Process Management Workshops, BPM 2006 International Workshops, BPD, BPI, ENEL, GPWW, DPM, semantics4ws, Vienna, Austria, September 4-7, 2006, Proceedings, volume 4103 of *Lecture Notes in Computer Science*, Springer, 2006, pp. 169–180. doi:10.1007/11837862_18.