Temporal Logic Specification-Conditioned Decision Transformer for Offline Safe Reinforcement Learning

Zijian Guo [1] **Weichao Zhou** [2] **Wenchao Li** [2]

Abstract

Offline safe reinforcement learning (RL) aims to train a constraint satisfaction policy from a fixed dataset. Current state-of-the-art approaches are based on supervised learning with a conditioned policy. However, these approaches fall short in real-world applications that involve complex tasks with rich temporal and logical structures. In this paper, we propose temporal logic Specification-conditioned Decision Transformer (SDT), a novel framework that harnesses the expressive power of signal temporal logic (STL) to specify complex temporal rules that an agent should follow and the sequential modeling capability of Decision Transformer (DT). Empirical evaluations on the `DSRL` benchmarks demonstrate the better capacity of SDT in learning safe and high-reward policies compared with existing approaches. In addition, SDT shows good alignment with respect to different desired degrees of satisfaction of the STL specification that it is conditioned on.

1. Introduction

Offline safe reinforcement learning (RL) is the problem of learning a policy to achieve high rewards while keeping the cost of constraint violation below a specified threshold from a fixed dataset without further interactions with the RL environment. It is preferable to online safe RL when data collection is expensive and inefficient (Gu et al., 2022b; Guo et al., 2023). Various methods have been proposed and shown promising performance in attaining high rewards and satisfying relatively simple constraints in applications such as autonomous driving (Gu et al., 2022a; Lin et al., 2023), robotics (Brunke et al., 2022), and healthcare (Kondrup et al., 2023; Zhang et al., 2024). However, real-world tasks often require agents to follow complex temporal and logical constraints. For example, autonomous-driving vehicles must come to a complete halt at an intersection with a stop sign, pause briefly, and proceed through it only if no other cars are present. It is challenging to define cost functions that appropriately capture such constraints.

Temporal logic (TL), on the other hand, provides a formalism for expressing the behaviors of systems over time, such as *safety* (e.g. never visit a bad state), *liveness* (e.g. eventually visit a good state), *sequentiality* (e.g. visit state A before visiting state B), and their arbitrarily elaborate combinations (Donzé, 2013; Tabuada & Neider, 2015; Coogan et al., 2017; Vasile et al., 2017; Madsen et al., 2018). Signal temporal logic (STL) is a variant of TL that can be used to specify properties over dense-time real-valued signals such as trajectories (Maler & Nickovic, 2004). A unique characteristic of STL is that it admits a quantitative semantics – a *robustness value* that quantifies the degree to which a given trajectory satisfies an STL formula (Fainekos & Pappas, 2009; Donzé & Maler, 2010). Due to this quantitative semantics and its expressiveness, STL has found a wide range of applications ranging from controller verification and synthesis (Eddeland et al., 2017; Sahin et al., 2020; Kurtz & Lin, 2021; Dawson & Fan, 2022; Zhang & Haesaert, 2023) to reinforcement learning (Balakrishnan & Deshmukh, 2019; Bozkurt et al., 2020; Zhang et al., 2021).

In this paper, we propose to leverage STL to solve offline safe RL problems that involve complex temporal constraints. To tackle the challenges of learning from a fixed dataset, we build on the Decision Transformer (DT) model (Chen et al., 2021) which leverages the self-attention mechanism in Transformers (Vaswani et al., 2017) to handle long-range dependencies and has demonstrated competitive performance in offline RL settings. A crucial insight of our work is that, while the notions of goals, returns, and costs are tied to RL, the sequence modeling and generation framework of DT is more general. Our novel framework, called temporal logic Specification-conditioned Decision Transformer (SDT), judiciously combines the sequential modeling capability of DT with the expressive power of STL to learn high-performance and safe policies. Our main contributions are summarized as follows.

[1]Division of Systems Engineering, Boston University [2]Department of Electrical and Computer Engineering, Boston University. Correspondence to: Zijian Guo <zjguo@bu.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

- We study the offline safe RL problem from the supervised learning perspective and propose SDT which enables conditioning on STL specifications in DT. SDT is the first work that incorporates STL to satisfy temporal constraints in offline safe RL settings.
- We examine the capacity of autoregressive learning with the quantitative semantics of STL. Our method introduces two key input tokens: the prefix and suffix robustness values that leverage different portions of a trajectory to provide complementary information.
- Our comprehensive experiments show that i) SDT outperforms multiple baselines both in safety and task performance by a large margin; ii) SDT can generalize to different robustness value thresholds and configurations without re-training the policy.

2. Related Work

Constrained optimization for offline safe RL. Offline safe RL is the intersection of safe RL (Achiam et al., 2017; Xu et al., 2021) and offline RL (Levine et al., 2020; Prudencio et al., 2023), where agents aim to balance safety and efficiency with reward, cost, and cost threshold information by learning from fixed trajectories. Most of the recent works formulate the safe RL problem as a constrained optimization problem. The Distribution Correction Estimation (DICE) family of RL algorithms (Kostrikov et al., 2021; Lee et al., 2021; 2022) optimizes the policy based on explicitly estimating the distributional shift between the target policy and the offline data distribution. The primal-dual method with function approximation optimizes iteratively between the policy and the lagrangian multiplier used to penalize constraint violations (Le et al., 2019; Chen et al., 2022; Xu et al., 2022; Polosky et al., 2022; Hong et al., 2023). However, the aspect of addressing temporal constraints remains underexplored.

Conditioned RL. Reward-Conditioned Supervised Learning (RCSL) represents a burgeoning category of algorithms that learn action distribution based on future return statistics via supervised learning. This idea is first proposed for the online RL setting (Schmidhuber, 2019; Kumar et al., 2019b; Peng et al., 2019). Decision Transformer (DT) and its variants (Chen et al., 2021; Furuta et al., 2022; Zheng et al., 2022; Yamagata et al., 2023; Wang et al., 2023; Hu et al., 2023) extend this idea to the offline RL setting by using return-to-go, i.e., cumulative future reward, as the conditional inputs and modeling trajectories with casual transformers (Vaswani et al., 2017). Instead of complex models, recent findings show that with careful policy tuning, simple multi-layered neural networks can match transformers' performance (Emmons et al., 2021; Brandfonbrener et al., 2022). Several works extend reward-conditioned RL methods in the offline safe RL setting by additionally conditioning on cost-to-go, i.e., cumulative future cost, and achieve not only safety and performance but also generalization to different cost thresholds (Liu et al., 2023b; Zhang et al., 2023). Our work adopts formal specifications in the conditions to characterize temporal properties that are difficult to be captured by standard reward or cost functions.

STL as reward (or cost) functions in RL. A well-defined reward (or cost) function is essential for RL agents to accomplish the underlying tasks (Amodei et al., 2016; Zhou & Li, 2022). Due to its rich syntax, STL can be used to specify high-level goals and safety requirements in RL tasks and also evaluate the behaviors of agents in conforming to the specifications combined with model-free methods (Li et al., 2017; Toro Icarte et al., 2018; Camacho et al., 2019; Balakrishnan & Deshmukh, 2019). Several model-based methods have been proposed (Kapoor et al., 2020; Cohen & Belta, 2021) to improve sample efficiency and promote safe training, but the inherent trial-and-error process of RL still poses a significant risk of performing unsafe actions. This work focuses on the offline setting to avoid unsafe explorations during training.

3. Preliminaries

3.1. Offline Safe RL

We consider learning in a finite horizon Markov decision process (MDP) which can be described by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the transition function and r is the reward function. A trajectory comprises a sequence of states and actions $\tau = \{s_t, a_t\}_{t=1}^{T}$ with length $|\tau| = T$. The goal of RL is to learn a policy π that maximizes the expected cumulative reward $\mathbb{E}_{\tau\sim\pi}[\sum_{t=1}^{T} r(s_t, a_t)]$.

A Constrained MDP (CMDP) (Altman, 2021) augments MDP with a cost function c, and is commonly used to model safe RL whose goal is to maximize the cumulative reward while limiting the cumulative cost to a threshold d:

$$\max_{\pi} \mathbb{E}_{\tau\sim\pi}\big[\sum_{t=1}^{T} r(s_t, a_t)\big], \text{ s.t. } \mathbb{E}_{\tau\sim\pi}\big[\sum_{t=1}^{T} c(s_t, a_t)\big] \leq d. \quad (1)$$

In the offline setting, the agent cannot interact with the environment and can only access a fixed dataset $\mathcal{D} = \{\tau_i\}_{i=1}^{n}$ consisting of trajectories collected by unknown policies.

3.2. Decision Transformers

The Decision Transformer (DT) model (Chen et al., 2021) tackles offline RL as a sequential modeling problem. Unlike the majority of prior RL approaches that estimate value functions and parameterize a single state-conditioned policy $\pi(a|s)$, DT outputs predicted actions from a sequence of return-to-go $\mathbf{R}_{t-K:t} = \{R_{t-K}, ..., R_t\}$, where $R_t = \sum_{t'=t}^{T} r_t(s_{t'}, a_{t'})$ is the cumulative reward from time-

step t and K is the context length; states $\mathbf{s}_{t-K:t} = \{s_{t-K}, ..., s_t\}$; and actions $\mathbf{a}_{t-K:t} = \{a_{t-K}, ..., a_t\}$. A DT's policy is parametrized by the GPT architecture (Radford et al., 2018) with a causal self-attention mask. It generates a deterministic action $\pi_{DT}(R_{t-K:t}, s_{t-K:t}, a_{t-K:t})$ at each time-step t. The policy is trained by minimizing the loss between the predicted and ground-truth actions.

3.3. STL Specification

Signal Temporal Logic (STL) is a formal logic used to specify and monitor temporal properties of continuous signals (Donzé & Maler, 2010). In this paper, we consider states at discrete time-steps as signals under discrete-time sampling. The syntax of STL is as follows.

$$\begin{aligned}\phi := \ & \top \mid \mu_c \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \Rightarrow \psi \mid \\ & \mathbf{G}_{[t_1,t_2]}\phi \mid \mathbf{F}_{[t_1,t_2]}\phi \mid \phi \mathbf{U}_{[t_1,t_2]}\psi\end{aligned} \tag{2}$$

where μ_c is a predicate of the form $\mu(s) < c$ where $\mu(\cdot) : \mathcal{S} \to \mathbb{R}$ and $c \in \mathbb{R}$ is a constant; ϕ and ψ are STL formulas; $t_1, t_2 \in \mathbb{Z}^+$ denote two sequential time steps. $\top$ represents True. The temporal operators $\mathbf{G}$, $\mathbf{F}$, and $\mathbf{U}$ refer to *Globally* (i.e., always), *Finally* (i.e., eventually), and *Until*.

The *quantitative semantics* (Donzé & Maler, 2010) of STL as shown in Eq. (3) quantifies the degree to which a trajectory τ satisfies or violates an STL formula ϕ at each time step t through the robustness value, denoted as $\rho(\tau, t, \phi)$.

$$\begin{aligned}\rho(\tau, t, \top) &= \rho_{max} \quad \text{where } \rho_{max} > 0 \\ \rho(\tau, t, \mu_c) &= c - \mu(s_t) \\ \rho(\tau, t, \neg\phi) &= -\rho(\tau, t, \phi) \\ \rho(\tau, t, \phi_1 \wedge \phi_2) &= \min\big(\rho(\tau, t, \phi_1), \rho(\tau, t, \phi_2)\big) \\ \rho(\tau, t, \phi_1 \vee \phi_2) &= \max\big(\rho(\tau, t, \phi_1), \rho(\tau, t, \phi_2)\big) \\ \rho(\tau, t, \phi \Rightarrow \psi) &= \max\big(-\rho(\tau, t, \phi), \rho(\tau, t, \psi)\big) \\ \rho(\tau, t, \mathbf{G}_{[t_1,t_2]}\phi) &= \min_{t' \in [t+t_1, t+t_2]} \big(\rho(\tau, t', \phi)\big) \\ \rho(\tau, t, \mathbf{F}_{[t_1,t_2]}\phi) &= \max_{t' \in [t+t_1, t+t_2]} \big(\rho(\tau, t', \phi)\big) \\ \rho(\tau, t, \phi \mathbf{U}_{[t_1,t_2]}\psi) &= \max_{t' \in [t+t_1, t+t_2]} \min\big(\rho(\tau, t', \psi), \\ & \qquad \min_{t'' \in [t, t']} \rho(\tau, t'', \phi)\big)\end{aligned} \tag{3}$$

The sign of the robustness value indicates whether the specification is satisfied or not, while a higher value indicates stronger satisfaction and vice versa. A specification can contain multiple predicates whose values can be arbitrarily scaled, yet the value of the overall specification maintains strict semantics.

4. Method

In this section, we first outline the formulation to solve the offline safe RL problem through supervised learning. Then, we present our novel framework, temporal logic Specification-conditioned Decision Transformer (SDT), and explain the rationale behind our specific approach of incorporating STL specification in DT.

4.1. Offline RL via Supervised Learning

Motivated by RCSL (Emmons et al., 2021; Brandfonbrener et al., 2022), we extend the supervised learning formulation to the offline safe RL setting. Instead of solving the constrained optimization problem in Eq. (1), the objective is to find an autoregressive model to simulate sampled trajectories conditioned on both reward and cost:

$$\max_{\theta} \mathbb{E}_{\mathbf{s} \sim \tau, \tau \sim \mathcal{D}} \Big[\log \pi_\theta\big(a | \mathbf{s}, r(\tau), c(\tau)\big)\Big] + \mathcal{L}_{\text{reg}} \tag{4}$$

where $\mathcal{L}_{\text{reg}}$ is the regularization term; $\mathbf{s} = \{s_{t-K}, ..., s_t\}$ is a sequence of states with a context length K; $r(\tau)$ and $c(\tau)$ represent the rewards and costs of the trajectory. This formulation captures a range of methods, as detailed in Table 1. RvS (Emmons et al., 2021) uses MLP policy with $K = 1$ and uses the $r(\tau)$ function to compute either the average future reward or future goal states. DT (Chen et al., 2021) adopts a transformers-based policy with $K > 1$ and uses $r(\tau)$ to specify return-to-go. ODT (Zheng et al., 2022) and QDT (Yamagata et al., 2023) follow similar principles as DT, but ODT incorporates stochastic policies and entropy regularization, and QDT learns Q-functions to relabel return-to-go. CDT (Liu et al., 2023b) extends to offline safe RL, utilizing cost-to-go in the $c(\tau)$ function. SaFormer (Zhang et al., 2023) estimates the cost-to-go to filter unsafe actions. Despite the variety, these existing methods center on Markovian rewards and costs. Our approach combines non-Markovian policies (transformers) with non-Markovian costs (robustness values of STL specifications).

Method	Architecture	r	c	$\mathcal{L}_{\text{reg}}$
RvS-R (Emmons et al., 2021)	MLP	average reward	–	–
RvS-G (Emmons et al., 2021)	MLP	goal state	–	–
DT (Chen et al., 2021)	Transformers	return-to-go	–	–
ODT (Zheng et al., 2022)	Transformers	return-to-go	–	Entropy
QDT (Yamagata et al., 2023)	Transformers	relabeled return-to-go	–	–
CDT (Liu et al., 2023b)	Transformers	return-to-go	cost-to-go	Entropy
SaFormer (Zhang et al., 2023)	Transformers	return-to-go	estimated cost-to-go	–
SDT(ours)	Transformers	return-go-to	robustness values	Entropy

Table 1: A brief comparison of methods with different architectures conditioned on different types of inputs. r (or c): reward (or cost) function.

4.2. Specification-Conditioned Decision Transformers

Our framework, named temporal logic Specification-conditioned Decision Transformer (SDT), as illustrated in Figure 1, extends the Decision Transformer (DT)

model (Chen et al., 2021) by incorporating two extra robustness value tokens, prefix and suffix, as defined in Definition 4.1. The intuition is to generate actions conditioned on both the reward and the trajectory's robustness in satisfying the specification. In this work, we focus on safety specifications[1] and preserve the reward token to measure performance. We employ a stochastic policy with entropy regularization, which has been shown to be effective in the literature (Zheng et al., 2022; Liu et al., 2023b).

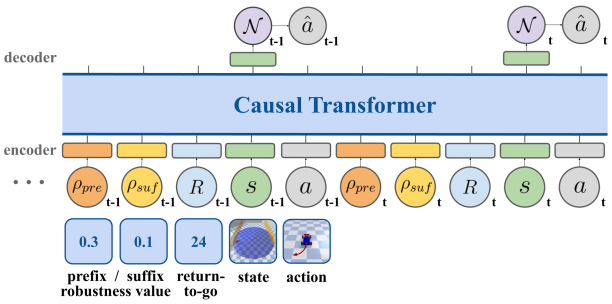

Figure 1: The SDT framework. It takes the prefix and suffix robustness values, return-to-go, states, and actions as inputs and predicts the next actions using a Gaussian policy.

Definition 4.1. (Prefix and suffix robustness value) Given a trajectory $\tau = \{s_1, s_2, ..., s_T\}$ with length $|\tau| = T$ and a STL specification ϕ, the prefix and suffix robustness value at time-step t are $\rho_{pre}(\tau, t, \phi) := \rho(\tau_{1:t}, 1, \phi)$ and $\rho_{suf}(\tau, t, \phi) := \rho(\tau_{t:T}, 1, \phi)$ respectively. For convenience, we will abbreviate "prefix and suffix robustness values" as "prefix" and "suffix" respectively.

Specifically, the input sequence $\mathbf{o}_t$ for SDT at time-step t includes prefix $\mathbf{P}_{pre} = \{\rho_{pre}(\tau, t-K, \phi), ..., \rho_{pre}(\tau, t, \phi)\}$, suffix $\mathbf{P}_{suf} = \{\rho_{suf}(\tau, t-K, \phi), ..., \rho_{suf}(\tau, t, \phi)\}$, rewards $\mathbf{R}_{t-K:t}$, states $\mathbf{s}_{t-K:t}$, and actions $\mathbf{a}_{t-K:t}$; the Gaussian policy parameterized by θ is $\pi_\theta(\cdot|\mathbf{o}_t) = \mathcal{N}(\mu_\theta(\mathbf{o}_t), \Sigma_\theta(\mathbf{o}_t))$; and the Shannon entropy regularizer is $\mathcal{L}_{\text{reg}} = H[\pi_\theta(\cdot|\mathbf{o})]$ with weight $\lambda \in [0, \infty)$. After plugging them into Eq. (4), the objective of SDT is:

$$\max_\theta \quad \mathbb{E}_{\mathbf{o}\sim\tau, \tau\sim\mathcal{D}}\left[\log \pi_\theta(\mathbf{a}|\mathbf{o}) + \lambda H[\pi_\theta(\cdot|\mathbf{o})]\right] \quad (5)$$

Training and evaluation. SDT generally follows the training and evaluation schemes of RCSL (Emmons et al., 2021; Brandfonbrener et al., 2022). The training procedure is as follows: sample a batch of sequences $\{\mathbf{o}, \mathbf{a}\}$ from the offline dataset $\mathcal{D}$, and then compute the loss in Eq. (5) to optimize the policy π_θ via gradient descent. In terms of evaluation, the procedure for SDT is presented in Algorithm 1. Updates to the return-to-go and the prefix occur in response to new rewards and state information from the environment. The target suffix at each time-step is specified instead of autoregressively computed, as detailed in the following section where we justify the usage of the suffix and prefix.

[1]In formal logic parlance, all specifications over finite trajectories are safety specifications. We use the term "safety" here to differentiate safety constraints from performance goals.

Algorithm 1 Evaluation procedure for SDT

Input: trained Transformer policy π_θ, STL specification ϕ, episode length T, context length K, target reward R and target suffix $\mathbf{P}_{suf}$, environment env

Get the initial state: $s_1 \leftarrow env.reset()$ and initial action: $a_t \leftarrow 0$ and initial input sequence: $\mathbf{o} \leftarrow \emptyset$.
for $t = 1, ..., T$ **do**
  Compute prefix $\rho_{pre} = \rho(s_{1:t}, \phi)$
  Set target suffix $\rho_{suf} = \mathbf{P}_{suf}[t]$
  Construct input sequence $\mathbf{o} \leftarrow \mathbf{o} \cap \{\rho_{suf}, \rho_{pre}, R_t, s_t, a_t\}$
  Get predicted action $\hat{a}_t \sim \pi(\cdot|\mathbf{o}_{t-K+1:t})$
  Execute the action: $s_{t+1}, r_t \leftarrow env.step(\hat{a}_t)$
  Compute target returns for the next step $R_{t+1} = R_t - r_t$ and set $a_t \leftarrow \hat{a}_t$
end for

4.3. Suffix and Prefix Robustness Values

Suffix robustness value as desired future. Recall that given a trajectory τ of length T, the return-to-go (or cost-to-go), at time-step t is computed by $R_t = \sum_{t'=t}^{T} r(s_{t'}, a_{t'})$ (or $C_t = \sum_{t'=t}^{T} c(s_{t'}, a_{t'})$). During training, they are concatenated with a sequence of states and actions and then fed to DT to optimize Eq. (4). (Furuta et al., 2022) point out that the conditional supervised learning approaches are performing *hindsight information matching*: match the output trajectories with future *information statistics* $I(\tau)$ to search for optimal actions. We introduce the suffix robustness value in Definition 4.1, which can be viewed as a particular form of $I(\tau)$, as an alternative to cost-to-go. For example, given a trajectory τ and a simple specification $\phi_1 = \mathbf{F}_{[1,10]}(\mathbf{s} > 0)$ describing that the states in a trajectory τ should *eventually* be greater than 0 within the next 10 steps. The corresponding suffix at time-step t can be computed as $\rho_{suf}(\tau, t, \phi_1) = \max_{t' \in [t+1, t+10]} s_{t'}$. The suffix can capture the statistics of future states. Moreover, STL suffix is strictly more expressive than cost-to-go (or return-to-go), as one can define a predicate as the sum of cost (or negative sum of reward) and the resulting suffix will be equivalent to negative cost-to-go (or return-to-go), e.g., $\mu_0(s_t) := C_t < 0 \Rightarrow \rho(\tau, t, \mu_0) = -C_t$. However, challenges arise in training and evaluation due to the following sparsity issue and the updates of STL robustness values.

Sparsity. The compliance or violation of STL specifications of a trajectory might be determined by a relatively small number of critical states or intervals within the trajectory. As

highlighted by the quantitative semantics of STL in Eq. (3), the suffix, or even a sequence of suffix can offer sparse information on its corresponding states. For example, given the STL formula ϕ_1, there may exist a trajectory τ whose states do not satisfy $\mathbf{s} > 0$ until the end of or near the end of τ. Consequently, a large portion of the suffix robustness values $\{\rho_{suf}(\tau, 1, \phi), ..., \rho_{suf}(\tau, T, \phi)\}$ equal to a constant. In this case, the agent cannot gain meaningful information due to limited feedback on robustness values.

Updates of STL robustness values. The cost-to-go (or return-to-go) can be updated autoregressively when a new cost (or reward) is obtained, i.e., $C_{t+1} = C_t - c(s_t, a_t)$ (or $R_{t+1} = R_t - r(s_t, a_t)$). However, the robustness value does not possess this additive property, i.e., $\rho_{suf}(\tau, t, \phi) \neq \rho_{suf}(\tau, t+1, \phi) - \rho(\tau_t, 1, \phi)$, Although the relation of two suffixes $\tau_{t:T}$ and $\tau_{t+1:T}$ of a given trajectory τ can be expressed as $\rho_{suf}(\tau, t, \phi) = f(\rho_{suf}(\tau, t+1, \phi), \rho(\tau_t, 1, \phi))$, f is a complex recursive mapping that depends on the specification, i.e., nested $\min()/\max()$ operations (Donzé et al., 2013; Dokhanchi et al., 2014; Deshmukh et al., 2017). Thus, the target suffix for each time-step has to be set in advance instead of computing the target reward and cost in an autoregressive manner. Empirically, we test and validate several different target suffix configurations in Section 5.3.

Prefix robustness value as achieved past. To deal with the sparsity issue of the suffix robustness value, we introduce the prefix robustness value, derived from the states of a trajectory up to time-step t, supplementary to the suffix. The intuition is that different segments of the trajectory, i.e., $\tau_{1:t}$ and $\tau_{t:T}$, can provide complementary information on this trajectory. While the prefix itself is subject to sparsity, we argue that *the comparison between the prefix and the suffix offers additional information about how robust the action is in terms of specification satisfaction.* Suppose that the prefix and suffix of a trajectory in the offline dataset at time-step t are ρ_{pre} and ρ_{suf}. When both ρ_{pre} and ρ_{suf} are positive, it indicates safety in both previous and future actions. Conversely, negative values for both suggest unsafe actions throughout the trajectory. A positive ρ_{pre} and a negative ρ_{suf} implies that while past actions have been safe, a future action at a certain time-step t' will be unsafe. In contrast, a negative ρ_{pre} and a positive ρ_{suf} indicate unsafe past actions but safe future actions. These insights can extend beyond the context length, aiding the policy in inferring the safety of both past and future states and actions even if they fall outside the current context, which cannot be achieved by cost-to-go (or return-to-go). Therefore, the combination of the prefix and suffix addresses the sparsity issue by providing comprehensive information about the trajectory. The ablation study in Section 5.3 shows that including both the prefix and suffix token is not only crucial for our SDT to learn a safe and high-reward policy but also can improve the performance of the standard DT framework, which uses only return-to-go.

5. Experiment

We use the following environments and STL specifications to evaluate SDT and baseline approaches and aim to answer the following questions:

- Can SDT learn policies that satisfy a given STL specification from offline datasets?
- Can SDT align with different target suffixes?
- How important are the prefix and suffix inputs in SDT?
- What is the influence of different target suffix configurations on the performance of SDT?
- Is SDT robust to rescaling individual predicates?

Environments. The `Bullet-Safety-gym` (Gronauer, 2022) is a public benchmark that includes a variety of robot locomotion tasks commonly used in previous works (Achiam et al., 2017; Chow et al., 2019). We consider three specific environments, `Run`, `Circle`, and `Reach`, where different types of robots, `Ball, Car, Drone`, and `Ant`, are trained. In the `Run` environment, agents earn rewards for achieving high speeds between two boundaries but incur penalties if they cross the boundaries or exceed an agent-specific velocity threshold. In the `Circle` environment, agents are rewarded for moving in a circular pattern but are constrained within a safe region smaller than the radius of the target circle. In the `Reach` environment, besides performing the same task as in the `Circle` environment, agents have an additional task of reaching goals in sequence. This setup of rewards and costs creates a dual influence on the agents' behaviors, where rewards motivate specific actions and costs act as deterrents for those actions. More details of the environments can be found in Appendix A.

Temporal behaviors. To evaluate the capability of SDT to satisfy temporal requirements, we consider the following STL specifications.

$$\phi_{\text{run}} = \mathbf{G}\Big(\psi_{\text{bndry}} \wedge \big(\neg\psi_{\text{vel}} \Rightarrow \mathbf{F}_{[1,5]}\psi_{\text{vel}}\big)\Big) \tag{6}$$

$$\phi_{\text{circle}} = \mathbf{G}\Big(\neg\psi_{\text{bndry}} \Rightarrow \mathbf{F}_{[1,5]}\psi_{\text{bndry}}\Big) \tag{7}$$

$$\phi_{\text{reach}} = \phi_{\text{circle}} \wedge \mathbf{F}\Big(\neg\psi_{\text{goalB}}\mathbf{U}\psi_{\text{goalA}}\Big) \tag{8}$$

where $\psi_{\text{bndry}} : s_t < d_{\text{lim}}$ is the predicate for staying within the safety boundary denoted as d_{lim}, $\psi_{\text{vel}} : s_t < v_{\text{lim}}$ is the predicate for maintaining a safe velocity denoted as v_{lim}, and ψ_{goal}[2] is the specification for reaching a small square region (since the predicates are linear) near a goal position

[2] ψ_{goal} is defined as $-s_t < d - x \wedge s_t < x + d \wedge -s_t < d - y \wedge s_t < y + d$ with goal $= [x, y]$ and $\rho(\tau, t, \psi_{\text{goal}}) = \min(-s_t + x - d, s_t - x - d, -s_t + y - d, s_t - y - d)$.

s_{goal}. Note that the state of the agent s_t contains the position and velocity information. In the `Run` environment, the agent is required to always stay between the boundaries, and if it exceeds the velocity threshold, it should slow down within the next 5 steps. In the `Circle` environment, the agent must leave the unsafe region within the next 5 steps once it enters the unsafe region. In the `Reach` environment, the agent must leave unsafe regions within a certain number of steps but also reach two goals in sequence, both located in the safe region, at least once. Although the specifications relax the original cost as it allows agents to enter the unsafe region as long as they can re-enter the safe region, they completely change the Markovian property of the original cost, making it challenging to learn safe policies. The corresponding robustness values of the prefix and suffix at time-step t can be calculated as:

$$\rho(\tau, t, \phi_{\text{run}}) = \min_{t' \in [t_1, t_2]} \Big(\min \Big(d_{\text{lim}} - s_{t'}, \max \big((v_{\text{lim}} - s_{t'}), \max_{t'' \in [t'+1, t'+5]} (v_{\text{lim}} - s_{t''}) \big) \Big) \Big) \tag{9}$$

$$\rho(\tau, t, \phi_{\text{circle}}) = \min_{t' \in [t_1, t_2]} \Big(\max \big((d_{\text{lim}} - s_{t'}), \max_{t'' \in [t'+1, t'+5]} (d_{\text{lim}} - s_{t''}) \big) \Big) \tag{10}$$

$$\rho(\tau, t, \phi_{\text{reach}}) = \min \Big(\rho(\tau, t, \phi_{\text{circle}}), \max_{t' \in [t_1, t_2]} \Big(\max_{t'' \in [t', t_2]} \min \big(\rho(\tau, t'', \psi_{\text{goalA}}), \min_{t''' \in [t', t'']} \rho(\tau, t''', \neg \psi_{\text{goalB}}) \big) \Big) \Big) \tag{11}$$

where $I = [t_1, t_2]$ is the interval. By setting I to be $[1, t]$ or $[t, T]$, we can obtain the prefix and suffix, respectively.

Offline dataset. We use the dataset from `DSRL` (Liu et al., 2023a), a comprehensive benchmark specialized for offline safe RL. Note that although this dataset contains the behaviors defined in the STL specifications, it is not designed for tasks with temporal constraints. For `Run` and `Circle` environments, to be consistent with the STL specifications, we define a new cost function to relabel the original cost c in the dataset:

$$\mathrm{c}'_t = \begin{cases} 1 & \text{if } c_{p_t} = 1 \text{ or } [c_{v_{t-5}}, ..., c_{v_{t-1}}] = \mathbf{1} \text{ for } \texttt{Run} \text{ envs} \\ 1 & \text{if } [c_{p_{t-5}}, ..., c_{p_{t-1}}] = \mathbf{1} \text{ for } \texttt{Circle} \text{ envs} \\ 0 & \text{otherwise} \end{cases}$$

where c'_t is the relabeled cost at time-step t and c_{p_t} and c_{v_t} denote the costs related to the position and velocity of agents, respectively, in the original dataset. The cumulative relabeled cost captures the violations against an STL specification, i.e., $C'_t = \sum_1^T c'_t = 0 \Leftrightarrow \rho(\tau_{1:T}, T, \phi) > 0$ for a trajectory $\tau = \{s_t, a_t, r_t, c'_t\}_{t=1}^T$. The relabeled cost-reward plots and suffix-reward plots are presented in Appendix A. For `Reach` environment, it is challenging to devise a cost function to capture the constraint precisely, i.e. relabel the cost in the offline dataset. Thus, we only train and evaluate the baselines that do not require costs. Moreover, it highlights the benefits of STL and the versatility of our method, which is applicable in broader scenarios, including both Markovian and non-Markovian constraints.

Metrics. Our evaluation metrics include (i) normalized cumulative reward, (ii) cumulative relabeled cost, which are consistent with the offline RL literature (Fu et al., 2020; Liu et al., 2023a), and (iii) the satisfaction rate that indicates the ratio of episodes in which the STL specification is satisfied within a maximum number of time-steps. We use the actual cumulative relabeled cost instead of the normalized one since $d = 0$. The normalized cumulative reward is:

$$R_{\text{normalized}} = \frac{R_\pi - r_{\min}(\mathcal{D})}{r_{\max}(\mathcal{D}) - r_{\min}(\mathcal{D})}$$

where R_π is the evaluated cumulative reward of policy π and $r_{\max}(\mathcal{D})$ and $r_{\min}(\mathcal{D})$ are the maximum cumulative reward and the minimum cumulative reward of the trajectories that satisfy the cost threshold in dataset $\mathcal{D}$. For convenience, we will abbreviate "normalized cumulative reward" as "reward" and "cumulative relabeled cost" as "cost".

Baselines. We present our results by comparing SDT with two categories of baselines: (i) constrained optimization methods and (ii) conditioned RL methods.

For the constrained optimization RL baselines, we consider two Lagrangian-based methods that use adaptive PID-based Lagrangian multipliers (Stooke et al., 2020) to penalize constraint violations: BCQ-Lagrangian (BCQ-Lag) and BEAR-Lagrangian (BEAR-Lag), which is built upon BCQ (Fujimoto et al., 2018) and BEAR (Kumar et al., 2019a), respectively. We also include CPQ (Xu et al., 2022), designed to learn safe policy by conservative cost estimations, and CoptiDICE (Lee et al., 2022), which learns safe policies via stationary distribution correction.

For the conditioned RL baselines, we consider CDT (Liu et al., 2023b) and construct two variants of the reward-conditioned RvS-R (Emmons et al., 2021): RvS-RC with an additional cost token and RvS-Rρ with two additional prefix and suffix tokens. We also include a Behavior Cloning baseline (BC-Safe) that only uses trajectories that satisfy the specifications to train the policy. Following the safe RL setting criteria (Ray et al., 2019), we prioritize safety: a policy that maintains a high satisfaction rate is preferred over ones that do not. The methods with a higher satisfaction rate than BC-safe are considered safe since strict zero-constraint violation is difficult if not impossible in a model-free setting. We use a small, fixed target suffix above zero at each time-step for methods trained on robustness values, except for the ablation study on SDT about target suffix configuration.

5.1. Can SDT learn specification-satisfying policies?

The evaluation results for different trained policies are presented in Table 2.These results reflect the average perfor-

Methods	Run-Average			Circle-Average			Reach-Average	
	Reward ↑	Cost ↓	Rate ↑	Reward ↑	Cost ↓	Rate ↑	Reward ↑	Rate ↑
SDT(ours)	**0.96±0.01**	0.33±2.04	**0.97±0.04**	0.89±0.08	**0.94±3.73**	**0.86±0.1**	**0.75 ± 0.13**	**0.76 ± 0.19**
RvS-Rρ	0.79±0.28	8.89±20.28	0.79±0.32	0.81±0.11	1.09±3.19	0.8±0.14	0.55 ± 0.25	0.65 ± 0.22
CDT	**0.96±0.03**	**0.32±1.62**	0.92±0.12	**0.92±0.03**	2.23±4.2	0.49±0.09	-	-
RvS-RC	1.27±0.63	20.34±26.69	0.53±0.41	0.75±0.26	9.65±29.64	0.62±0.1	-	-
BC-safe	0.78±0.29	3.18±8.5	0.78±0.25	0.62±0.28	5.75±10.51	0.49±0.13	0.58 ± 0.3	0.34 ± 0.15
BCQ-Lag	0.9±0.3	30.08±39.49	0.3±0.36	0.99±0.34	33.32±21.25	0.08±0.13	-	-
BEAR-Lag	0.85±1.16	60.3±46.47	0.33±0.47	1.07±0.18	33.2±18.3	0.0±0.0	-	-
CPQ	1.16±1.33	50.29±48.38	0.33±0.47	0.59±0.35	5.73±18.94	0.72±0.36	-	-
COptiDICE	0.9±0.13	21.83±27.68	0.41±0.46	0.66±0.13	12.41±17.57	0.17±0.13	-	-

Table 2: Evaluation results of reward, cost, and satisfaction rate. ↑: the higher the reward, the better. ↓: the lower the cost (closer to 0), the better. Agents with a higher satisfaction rate than BC-safe are considered safe. Gray: unsafe agents. **Bold**: the best (highest reward, highest specification satisfaction rate, and lowest cost) in the respective metric. The satisfaction rates of the trajectories in the offline dataset are 20.0%, 8.52%, and 3.47% for `Run`, `Circle`, and `Reach` environments. For `Reach` environment, it is challenging to devise a cost function to capture the constraint precisely, i.e. relabel the cost in the offline dataset. Thus, we only train and evaluate the baselines that use robustness values for this environment.

mance, with each plot aggregating data from 3 random seeds and 20 trajectories per seed. Complete results for each environment are included in AppendixB due to page limit. *Our method demonstrates the best performance compared to the baselines in terms of reaching the highest rate of specification satisfaction and lowest costs*. Also, it produces comparable or higher rewards compared to the safe baselines, indicating that SDT performs effective learning.

The results of CDT and RvS-RC show that merely relabeling the cost to reflect STL specifications cannot obtain safe policies. Such cost function introduces a level of stochasticity (Paster et al., 2022), where taking the same action in the same states can lead to inconsistent cost due to the temporal constraint. Note that in the `Run` environments, RvS-RC even outperforms the best safe trajectory's reward in the dataset; however, the satisfaction rate is low and the cost is high. This pattern, observed in most of the baselines, where a high reward often correlates with a high cost, underscores the inherent trade-off between rewards and costs (Liu et al., 2022; Li et al., 2023). RvS-Rρ shows improved results over RvS-RC, as it benefits from training with the prefix and suffix. Our method still outperforms it due to the sequential structure of transformers, which is adept at capturing the temporal properties in robustness values, states, and actions.

The Lagrangian-based baselines, BCQ-Lag and BEAR-Lag, as well as CPQ and COptiDICE methods that are tailored for offline safe RL, struggle to ensure safety in most environments. This indicates that directly applying widely used safe RL techniques to satisfy temporal specifications does not work well. A key factor for the poor performance is the non-Markovian nature of the relabeled cost since these methods are designed to estimate value functions based on Markovian costs. As the reward is unchanged in the dataset, the baselines manage to secure high rewards. We can also notice that the reward of CPQ in the `Run` environments and the reward of BEAR-Lag in the `Circle` environments exceed the maximum reward of the safe trajectories in the dataset, which shows the stitching property of the Q-learning-based methods (Yamagata et al., 2023; Wang et al., 2023). The poor safety performance of the baselines highlights the challenges posed by the temporal constraints. In comparison, our proposed SDT method successfully learns both safe and rewarding policies in these challenging environments.

5.2. Can SDT adapt to different target suffixes?

One of the critical attributes of transformers is the ability to align with the variables on which they are conditioned. To illustrate this, we vary the target reward and target suffix for evaluation rollouts and obtain the results in Figure 2. It is evident that the baselines introduced previously, except RvS, lack this capability because they depend on a constant and pre-defined threshold to solve a constrained optimization problem and need re-training for adaptation to new constraints. Therefore, our comparison is focused on SDT and RvS-Rρ. *The results show a strong correlation between the actual and target suffixes for SDT.* We can also observe that the actual suffix of SDT is above the dashed threshold line when the target suffix is less than zero in most of the environments, which means that SDT can achieve safer actions even under conditions that demand otherwise. While there is a saturation point in the curves at specific target suffixes, SDT consistently upholds safety even when extrapolating over target suffixes and rewards that represent

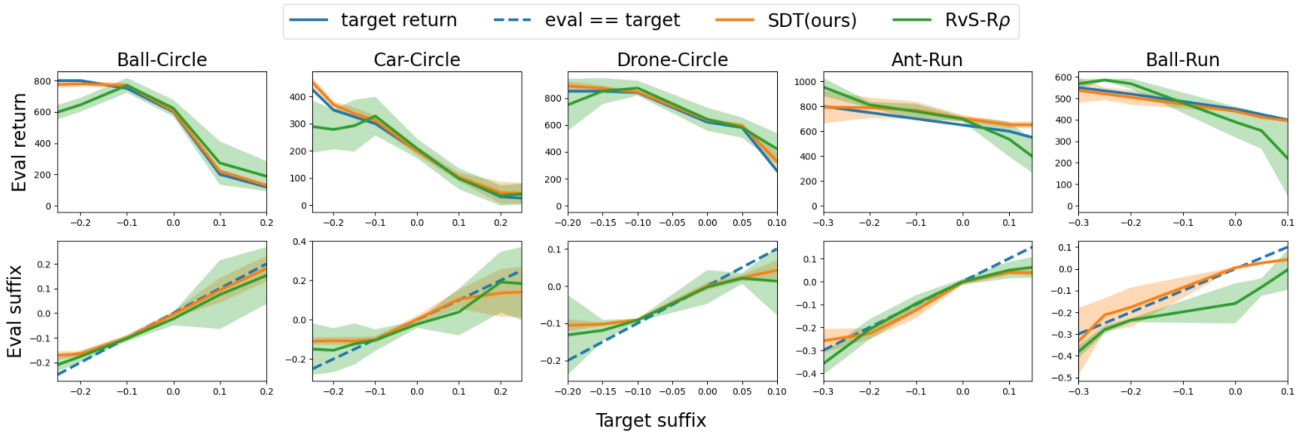

Figure 2: Results of alignment with different target suffixes. The top-row plots show the evaluated reward and the bottom-row plots show the evaluated suffix. The solid line and the light shade area represent the mean and mean $\pm$ standard deviation.

conflicting objectives. For example, setting both a high target reward and a high target suffix is unreachable since the former encourages agents to stay close to the safety boundary while the latter advises against it. In contrast, RvS-Rρ struggles to meet the target reward when the target suffix is either large or small, further showcasing the strong alignment capabilities of our method.

5.3. Ablation studies

How important are the prefix and suffix inputs in SDT? To assess the influence of the prefix and suffix, we conduct experiments by removing each of them from SDT. The left plot in Figure 4 shows the averaged performance of SDT, with the cost normalized against the highest cost among the corresponding experiments. The results indicate that both the prefix and suffix are essential for good performance, as their removal leads to noticeable drops in safety and performance. In addition, the suffix has a more pronounced effect on the safety performance than the prefix, evidenced by a significant reduction in the satisfaction rate when the suffix is excluded since the prefix lacks hindsight information.

Following the insights in Section 4.3 that the relation between prefix and suffix can provide additional information, we re-assess the efficacy of DT with an additional reward prefix token $R_{pre} = \sum_{i=1}^{t} r_i$ while keeping all the other aspects unchanged. More details about the experiments of DT can be found in Appendix C. From Figure 3, we can see that DT with reward prefix achieves higher rewards than the standard DT. Although the reward prefix is implicitly embedded in the return-to-go, i.e., given the return-to-go of a trajectory at each time-step, we can fully recover the reward prefix, explicitly supplying the past information to transformers shows advantageous for policy learning. On the other hand, we cannot extract prefix from suffix in general for STL robustness values, thus necessitating explicit inputs in SDT. Recent studies have explored how transformers process explicit inputs versus implicit inputs, with varying conclusions as it largely depends on the specific nature of the data and the task (Zhao et al., 2019; Chu et al., 2021; Gui et al., 2021). In our opinion, given enough capacity and data, transformers can learn effectively regardless of the input type, but in practice, transformers tend to learn better with explicit inputs.

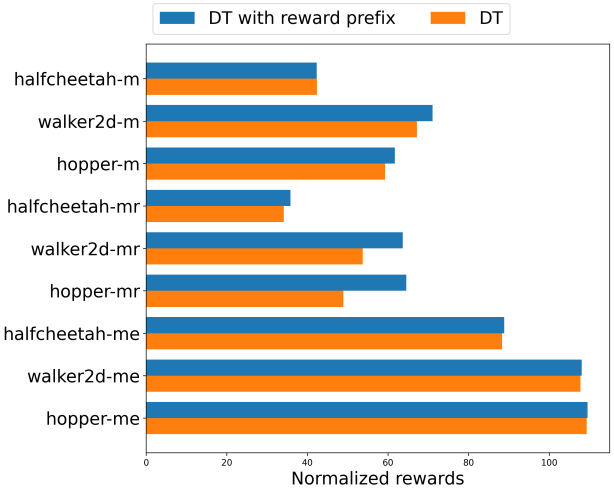

Figure 3: Evaluation results of normalized rewards in D4RL Gym environments. Each value is averaged over 3 seeds. m: medium, mr: medium-replay, me: medium-expert.

How do different target suffix configurations influence the performance of SDT? As mentioned in Section 4.2, the suffix offers versatility in evaluations, as we can specify the target suffix for each time-step of the trajectory instead of the auto-regressive manner of return-to-go (or cost-to-go). We test SDT using the same target reward but different target suffix configurations: i) SDT(linear), where the target suffix linearly increases each step until it matches the fixed

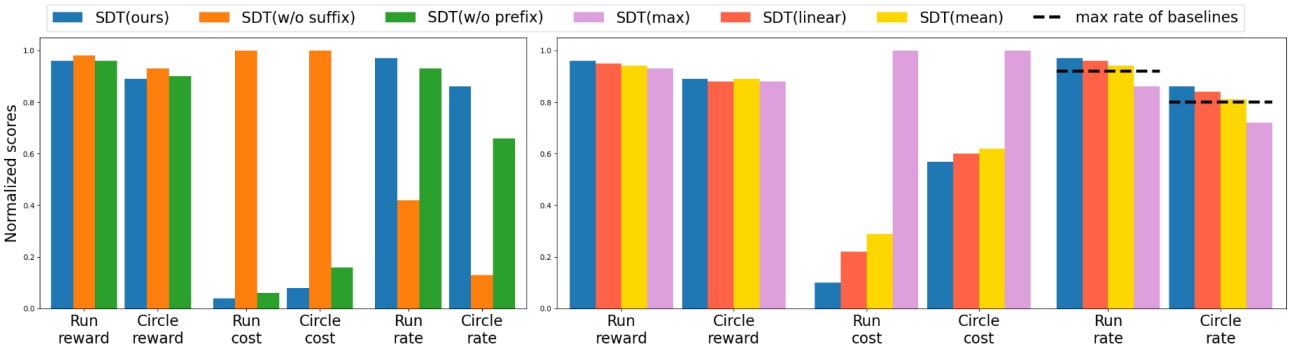

Figure 4: Ablation study. Left: the effect of the prefix and the suffix. Right: influence of different target suffix configurations.

target; ii) SDT(mean), employing the average suffix from safe trajectories in the dataset; iii) SDT(max), using the maximum suffix from these trajectories. SDT(linear) is selected as the suffix is monotonically increasing for trajectories in the dataset due to the $\mathbf{G}$ operation in the STL specification in Eq. (6) and (7). SDT(mean) aims to mimic the average performance of the safe trajectories, whereas SDT(max) is expected to behave conservatively by emulating the trajectory with the highest suffix. Details of the prefix and the suffix for dataset trajectories are in Appendix A. As shown in Figure 4, we can observe that SDT(ours) achieves the best safety performance with a fixed target suffix. Other target suffix configurations lead to reduced satisfaction rates and increased costs because: i) a low target suffix makes the agent more aggressive, i.e., SDT(linear), and ii) conflicting objectives of achieving a high target suffix and high target reward at the same time, i.e., SDT(mean) and SDT(max). However, it is worth noting that these suffix configurations still outperform the baselines in terms of satisfaction rate.

Methods	Reward	Rate
SDT($\alpha_b = 1, \alpha_v = 1$)	0.96 ± 0.01	0.97 ± 0.04
SDT($\alpha_b = 1, \alpha_v = 10$)	0.95 ± 0.02	0.97 ± 0.06
SDT($\alpha_b = 1, \alpha_v = 100$)	0.95 ± 0.03	0.99 ± 0.02
SDT($\alpha_b = 10, \alpha_v = 1$)	0.95 ± 0.02	0.96 ± 0.07
SDT($\alpha_b = 100, \alpha_v = 1$)	0.93 ± 0.03	0.98 ± 0.03

Table 3: Evaluation results of reward and satisfaction rate with varying scaling factors.

Is SDT robust to rescaling individual predicates? Though scaling individual predicates does not change the sign of a specification, in practice, it is possible that certain predicates are dominated if they are scaled down, leading to a violation of the specification. To evaluate the rescaling effect, we incorporate two scaling factors into Eq. (6):

$$\phi_{\text{run}} = \mathbf{G}\Big(\alpha_b \psi_{\text{bndry}} \wedge \big(\neg \alpha_v \psi_{\text{vel}} \Rightarrow \mathbf{F}_{[1,5]} \alpha_v \psi_{\text{vel}}\big)\Big) \quad (12)$$

We explore various pairings of α_b and α_v, and the results are shown in Table 3. Our findings demonstrate that modifying the scale of the predicates has a negligible impact on both the task performance and the rate of property satisfaction, which indicates that our method is robust to changes in predicate scaling.

6. Conclusion

We study the offline safe RL problem through the lens of supervised learning and point out the unique challenges associated with enforcing temporal constraints. We propose a novel framework that utilizes the robustness value of STL specifications to guide the trajectory modeling process in DT. Our empirical results demonstrate that SDT is capable of learning a safe and high-reward policy in challenging offline safe RL tasks that involve temporal and logical requirements, and can adapt to different target suffixes without re-training and performs effectively across diverse target suffix configurations. Future works will explore the use of STL to specify both safety and performance objectives in DT.

Acknowledgement

The authors thank the anonymous reviewers for their invaluable feedback and constructive suggestions. This material is based upon work supported by the National Science Foundation under Grant No. CCF-2340776.

Impact Statement

This paper presents a novel framework within the realm of reinforcement learning, aiming to advance the field through innovative approaches and applications. First of all, the methods, experiments, and results outlined in this paper do not pose any ethical concerns. Secondly, it's crucial for researchers to proceed with care, especially when setting specifications and conducting tests in real-world settings, since misspecified specifications may result in serious and unforeseen consequences. Lastly, we hope our findings can provide fresh insights for extending the application of reinforcement learning to broader domains.

References

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In *International conference on machine learning*, pp. 22–31. PMLR, 2017.

Altman, E. *Constrained Markov decision processes*. Routledge, 2021.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Balakrishnan, A. and Deshmukh, J. V. Structured reward shaping using signal temporal logic specifications. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3481–3486. IEEE, 2019.

Bozkurt, A. K., Wang, Y., Zavlanos, M. M., and Pajic, M. Control synthesis from linear temporal logic specifications using model-free reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10349–10355. IEEE, 2020.

Brandfonbrener, D., Bietti, A., Buckman, J., Laroche, R., and Bruna, J. When does return-conditioned supervised learning work for offline reinforcement learning? *Advances in Neural Information Processing Systems*, 35: 1542–1553, 2022.

Brunke, L., Greeff, M., Hall, A. W., Yuan, Z., Zhou, S., Panerati, J., and Schoellig, A. P. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:411–444, 2022.

Camacho, A., Icarte, R. T., Klassen, T. Q., Valenzano, R. A., and McIlraith, S. A. Ltl and beyond: Formal languages for reward function specification in reinforcement learning. In *IJCAI*, volume 19, pp. 6065–6073, 2019.

Chen, F., Zhang, J., and Wen, Z. A near-optimal primal-dual method for off-policy learning in cmdp. *Advances in Neural Information Processing Systems*, 35:10521–10532, 2022.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Chow, Y., Nachum, O., Faust, A., Duenez-Guzman, E., and Ghavamzadeh, M. Lyapunov-based safe policy optimization for continuous control. *arXiv preprint arXiv:1901.10031*, 2019.

Chu, X., Tian, Z., Zhang, B., Wang, X., Wei, X., Xia, H., and Shen, C. Conditional positional encodings for vision transformers. *arXiv preprint arXiv:2102.10882*, 2021.

Cohen, M. H. and Belta, C. Model-based reinforcement learning for approximate optimal control with temporal logic specifications. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, pp. 1–11, 2021.

Coogan, S., Arcak, M., and Belta, C. Formal methods for control of traffic flow: Automated control synthesis from finite-state transition models. *IEEE Control Systems Magazine*, 37(2):109–128, 2017.

Dawson, C. and Fan, C. Robust counterexample-guided optimization for planning from differentiable temporal logic. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7205–7212. IEEE, 2022.

Deshmukh, J. V., Donzé, A., Ghosh, S., Jin, X., Juniwal, G., and Seshia, S. A. Robust online monitoring of signal temporal logic. *Formal Methods in System Design*, 51: 5–30, 2017.

Dokhanchi, A., Hoxha, B., and Fainekos, G. On-line monitoring for temporal logic robustness. In *International Conference on Runtime Verification*, pp. 231–246. Springer, 2014.

Donzé, A. On signal temporal logic. In *Runtime Verification: 4th International Conference, RV 2013, Rennes, France, September 24-27, 2013. Proceedings 4*, pp. 382–383. Springer, 2013.

Donzé, A. and Maler, O. Robust satisfaction of temporal logic over real-valued signals. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pp. 92–106. Springer, 2010.

Donzé, A., Ferrere, T., and Maler, O. Efficient robust monitoring for stl. In *Computer Aided Verification: 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings 25*, pp. 264–279. Springer, 2013.

Eddeland, J., Miremadi, S., Fabian, M., and Åkesson, K. Objective functions for falsification of signal temporal logic properties in cyber-physical systems. In *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pp. 1326–1331. IEEE, 2017.

Emmons, S., Eysenbach, B., Kostrikov, I., and Levine, S. Rvs: What is essential for offline rl via supervised learning? *arXiv preprint arXiv:2112.10751*, 2021.

Fainekos, G. E. and Pappas, G. J. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, 2009.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. corr abs/1812.02900 (2018). *arXiv preprint arXiv:1812.02900*, 2018.

Furuta, H., Matsuo, Y., and Gu, S. S. Generalized decision transformer for offline hindsight information matching. In *International Conference on Learning Representations*, 2022.

Gronauer, S. Bullet-safety-gym: Aframework for constrained reinforcement learning. 2022.

Gu, S., Chen, G., Zhang, L., Hou, J., Hu, Y., and Knoll, A. Constrained reinforcement learning for vehicle motion planning with topological reachability analysis. *Robotics*, 11(4):81, 2022a.

Gu, S., Yang, L., Du, Y., Chen, G., Walter, F., Wang, J., Yang, Y., and Knoll, A. A review of safe reinforcement learning: Methods, theory and applications. *arXiv preprint arXiv:2205.10330*, 2022b.

Gui, L., Wang, B., Huang, Q., Hauptmann, A., Bisk, Y., and Gao, J. Kat: A knowledge augmented transformer for vision-and-language. *arXiv preprint arXiv:2112.08614*, 2021.

Guo, S., Zou, L., Chen, H., Qu, B., Chi, H., Philip, S. Y., and Chang, Y. Sample efficient offline-to-online reinforcement learning. *IEEE Transactions on Knowledge and Data Engineering*, 2023.

Hong, K., Li, Y., and Tewari, A. A primal-dual-critic algorithm for offline constrained reinforcement learning. *arXiv preprint arXiv:2306.07818*, 2023.

Hu, S., Shen, L., Zhang, Y., and Tao, D. Graph decision transformer. *arXiv preprint arXiv:2303.03747*, 2023.

Ji, J., Zhou, J., Zhang, B., Dai, J., Pan, X., Sun, R., Huang, W., Geng, Y., Liu, M., and Yang, Y. Omnisafe: An infrastructure for accelerating safe reinforcement learning research. *arXiv preprint arXiv:2305.09304*, 2023.

Kapoor, P., Balakrishnan, A., and Deshmukh, J. V. Model-based reinforcement learning from signal temporal logic specifications. *arXiv preprint arXiv:2011.04950*, 2020.

Kondrup, F., Jiralerspong, T., Lau, E., de Lara, N., Shkrob, J., Tran, M. D., Precup, D., and Basu, S. Towards safe mechanical ventilation treatment using deep offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 15696–15702, 2023.

Kostrikov, I., Fergus, R., Tompson, J., and Nachum, O. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*, pp. 5774–5783. PMLR, 2021.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019a.

Kumar, A., Peng, X. B., and Levine, S. Reward-conditioned policies. *arXiv preprint arXiv:1912.13465*, 2019b.

Kurtz, V. and Lin, H. A more scalable mixed-integer encoding for metric temporal logic. *IEEE Control Systems Letters*, 6:1718–1723, 2021.

Le, H., Voloshin, C., and Yue, Y. Batch policy learning under constraints. In *International Conference on Machine Learning*, pp. 3703–3712. PMLR, 2019.

Lee, J., Jeon, W., Lee, B., Pineau, J., and Kim, K.-E. Optidice: Offline policy optimization via stationary distribution correction estimation. In *International Conference on Machine Learning*, pp. 6120–6130. PMLR, 2021.

Lee, J., Paduraru, C., Mankowitz, D. J., Heess, N., Precup, D., Kim, K.-E., and Guez, A. Coptidice: Offline constrained reinforcement learning via stationary distribution correction estimation. *arXiv preprint arXiv:2204.08957*, 2022.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Li, J., Liu, X., Zhu, B., Jiao, J., Tomizuka, M., Tang, C., and Zhan, W. Guided online distillation: Promoting safe reinforcement learning by offline demonstration. *arXiv preprint arXiv:2309.09408*, 2023.

Li, X., Vasile, C.-I., and Belta, C. Reinforcement learning with temporal logic rewards. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3834–3839. IEEE, 2017.

Lin, H., Ding, W., Liu, Z., Niu, Y., Zhu, J., Niu, Y., and Zhao, D. Safety-aware causal representation for trustworthy reinforcement learning in autonomous driving. *arXiv preprint arXiv:2311.10747*, 2023.

Liu, Z., Guo, Z., Cen, Z., Zhang, H., Tan, J., Li, B., and Zhao, D. On the robustness of safe reinforcement learning under observational perturbations. *arXiv preprint arXiv:2205.14691*, 2022.

Liu, Z., Guo, Z., Lin, H., Yao, Y., Zhu, J., Cen, Z., Hu, H., Yu, W., Zhang, T., Tan, J., et al. Datasets and benchmarks for offline safe reinforcement learning. *arXiv preprint arXiv:2306.09303*, 2023a.

Liu, Z., Guo, Z., Yao, Y., Cen, Z., Yu, W., Zhang, T., and Zhao, D. Constrained decision transformer for offline safe reinforcement learning. *arXiv preprint arXiv:2302.07351*, 2023b.

Madsen, C., Vaidyanathan, P., Sadraddini, S., Vasile, C.-I., DeLateur, N. A., Weiss, R., Densmore, D., and Belta, C. Metrics for signal temporal logic formulae. In *2018 IEEE Conference on Decision and Control (CDC)*, pp. 1542–1547. IEEE, 2018.

Maler, O. and Nickovic, D. Monitoring temporal properties of continuous signals. In *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, pp. 152–166. Springer, 2004.

Paster, K., McIlraith, S., and Ba, J. You can't count on luck: Why decision transformers and rvs fail in stochastic environments. *Advances in Neural Information Processing Systems*, 35:38966–38979, 2022.

Peng, X. B., Kumar, A., Zhang, G., and Levine, S. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Polosky, N., Da Silva, B. C., Fiterau, M., and Jagannath, J. Constrained offline policy optimization. In *International Conference on Machine Learning*, pp. 17801–17810. PMLR, 2022.

Prudencio, R. F., Maximo, M. R., and Colombini, E. L. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. Improving language understanding by generative pre-training. 2018.

Ray, A., Achiam, J., and Amodei, D. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 7(1):2, 2019.

Sahin, Y. E., Quirynen, R., and Di Cairano, S. Autonomous vehicle decision-making and monitoring based on signal temporal logic and mixed-integer programming. In *2020 American Control Conference (ACC)*, pp. 454–459. IEEE, 2020.

Schmidhuber, J. Reinforcement learning upside down: Don't predict rewards–just map them to actions. *arXiv preprint arXiv:1912.02875*, 2019.

Stooke, A., Achiam, J., and Abbeel, P. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*, pp. 9133–9143. PMLR, 2020.

Tabuada, P. and Neider, D. Robust linear temporal logic. *arXiv preprint arXiv:1510.08970*, 2015.

Toro Icarte, R., Klassen, T. Q., Valenzano, R., and McIlraith, S. A. Teaching multiple tasks to an rl agent using ltl. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 452–461, 2018.

Vasile, C.-I., Tumova, J., Karaman, S., Belta, C., and Rus, D. Minimum-violation scltl motion planning for mobility-on-demand. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1481–1488. IEEE, 2017.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, Y., Yang, C., Wen, Y., Liu, Y., and Qiao, Y. Critic-guided decision transformer for offline reinforcement learning. *arXiv preprint arXiv:2312.13716*, 2023.

Xu, H., Zhan, X., and Zhu, X. Constraints penalized q-learning for safe offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8753–8760, 2022.

Xu, T., Liang, Y., and Lan, G. Crpo: A new approach for safe reinforcement learning with convergence guarantee. In *International Conference on Machine Learning*, pp. 11480–11491. PMLR, 2021.

Yamagata, T., Khalil, A., and Santos-Rodriguez, R. Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline rl. In *International Conference on Machine Learning*, pp. 38989–39007. PMLR, 2023.

Zhang, B., Qiu, X., and Tan, X. Balancing therapeutic effect and safety in ventilator parameter recommendation: An offline reinforcement learning approach. *Engineering Applications of Artificial Intelligence*, 131:107784, 2024.

Zhang, Q., Zhang, L., Xu, H., Shen, L., Wang, B., Chang, Y., Wang, X., Yuan, B., and Tao, D. Saformer: A conditional sequence modeling approach to offline safe reinforcement learning. *arXiv preprint arXiv:2301.12203*, 2023.

Zhang, X., Peng, Y., Luo, B., Pan, W., Xu, X., and Xie, H. Model-based safe reinforcement learning with time-varying state and control constraints: An application to intelligent vehicles. *arXiv preprint arXiv:2112.11217*, 2021.

Zhang, Z. and Haesaert, S. Modularized control synthesis for complex signal temporal logic specifications. *arXiv preprint arXiv:2303.17086*, 2023.

Zhao, G., Lin, J., Zhang, Z., Ren, X., Su, Q., and Sun, X. Explicit sparse transformer: Concentrated attention through explicit selection. *arXiv preprint arXiv:1912.11637*, 2019.

Zheng, Q., Zhang, A., and Grover, A. Online decision transformer. In *international conference on machine learning*, pp. 27042–27059. PMLR, 2022.

Zhou, W. and Li, W. Programmatic reward design by example. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8):9233–9241, Jun. 2022. doi: 10.1609/aaai.v36i8.20910.

A. Environment Setting

Reward and cost functions defined in the environments. We use the `Bullet-safety-gym` (Gronauer, 2022) environments for this set of experiments. In the `Run` environments, agents receive rewards for high-speed movement between two safety boundaries. However, they incur penalties when they either cross these boundaries or surpass a velocity threshold that is specific to different types of robots. The reward and cost function are defined as:

$$r(\boldsymbol{s_t}) = \frac{-y_t v_x + x_t v_y}{1 + ||\sqrt{x_t^2 + y_t^2} - r|}$$
$$c(\boldsymbol{s_t}) = \mathbf{1}(|x| > x_{\text{lim}})$$

where $\boldsymbol{s}_t = [x_t, y_t, v_x, v_y]$, r is the radius of the circle, and x_{lim} specifies the range of the safety region. In the `Circle` environments, agents gain rewards for circular motion in a clockwise direction but are required to remain inside a designated safe area, which is smaller than the circumference of the intended circle. The reward and cost functions are defined as:

$$r(\boldsymbol{s_t}) = ||\boldsymbol{x_{t-1}} - \boldsymbol{g}||_2 - ||\boldsymbol{x}_t - \boldsymbol{g}||_2$$
$$c(\boldsymbol{s_t}) = \mathbf{1}(|y| > y_{\text{lim}} \text{ or } ||\boldsymbol{v_t}||_2 > v_{\text{lim}})$$

where y_{lim} is the safety boundary and v_{lim} is the velocity limit. To establish the criteria that zero cost means no violations of the specification, we relabel the cost. For the `Circle` environment, the relabeled cost of the current state is 1 if the costs of its previous 5 steps are all 1. For the `Run` environment, the relabeled cost of the current state is 1 if the costs related to speeding of its previous 5 steps are all 1 or the cost related to safe boundary crossing is 1.

Offline dataset visualization. The dataset suffix-reward and relabeled cost-reward plots for the training tasks `Ant-Run`, `Ball-Run`, `Drone-Run`, `Ball-Circle`, `Car-Circle`, and `Drone-Circle`, are shown in Figure. 5. Analyzing the figures provided, we can generally discern an increasing trend for the reward in relation to the cost. In other words, as cost increases, so too might the reward return, underscoring the inherent trade-off between reward and cost. This phenomenon aligns with findings discussed in previous works (Liu et al., 2023b; 2022), which is also one of the reasons that SDT is evaluated in `Bullet-Safety-Gym` environments. In contrast, the same clear increasing trend is not observable in `SafetyGymnasium` environments (Ray et al., 2019; Liu et al., 2023a; Ji et al., 2023), such as `Goal`, `Button`, and `Push`. Moreover, highly stochastic environments pose unique challenges to the RCSL algorithms (Paster et al., 2022) and are beyond the scope of this paper.

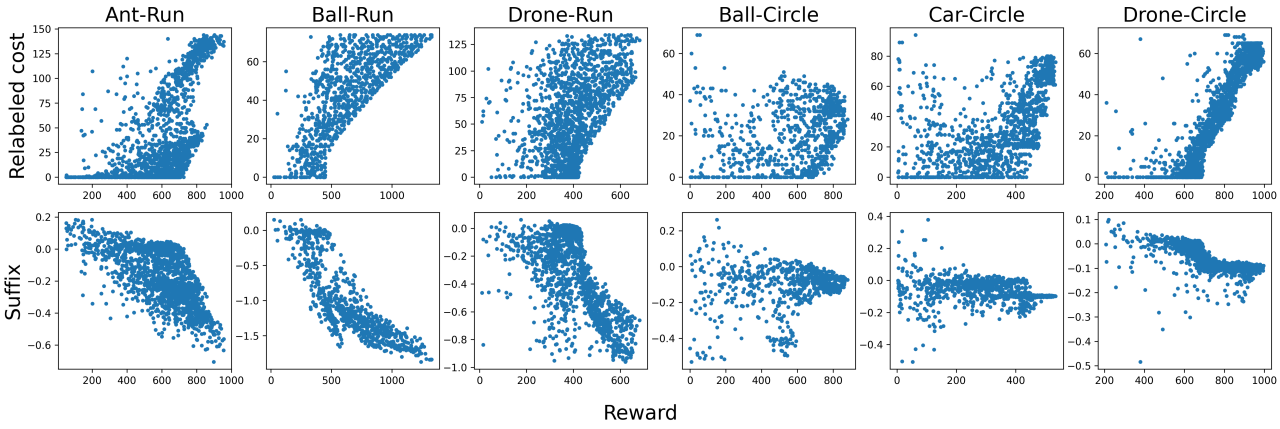

Figure 5: Illustration of the offline dataset. The first-row plots show the relabeled cost versus reward and the second-row plots show the suffix ($\rho_{suf}(\tau_{1:T}, 1, \phi)$) versus reward. Each column represents an environment. Each point denotes a collected trajectory (not necessarily to be unique) with corresponding episodic relabeled cost (or suffix) and reward value.

SDT and baselines implementation. Our implementation of SDT is built on the public codebase provided by the `OSRL` library[3], which offers a collection of elegant and extensible implementations of state-of-the-art offline safe RL algorithms. We use the `STLCG` toolbox[4] to compute the robustness value of the specification and add the corresponding prefix and suffix

[3]https://github.com/liuzuxin/OSRL
[4]https://github.com/StanfordASL/stlcg

tokens as input to the transformers. In our experiments, we train SDT and baselines 200000 steps to ensure convergence and keep the rest of the hyperparameters used in the library unchanged. The constrained optimization RL baselines are trained and evaluated using a cost threshold of $d = 0$. During evaluation, the target cost is 0 for CDT and RvS-RC. The target suffix is $\{0.02, 0.01, 0.02, 0.09, 0.06, 0.02, 0.06, 0.06, 0.04\}$ for SDT and RvS-Rρ and the target reward is $\{720, 440, 410, 610, 400, 650, 300, 300, 600\}$ for SDT and the conditioned RL baselines in `Ant-Run`, `Ball-Run`, `Drone-Run`, `Ball-Circle`, `Car-Circle`, `Drone-Circle`, `Ball-Reach`, `Car-Reach`, and `Drone-Reach`, respectively.

B. Complete Results of SDT

The full evaluation results for different trained policies are presented in Table 4. The columns of average performance are the same as Table 2. All values are averaged among 3 random seeds and 20 trajectories for each seed. SDT, CDT, RvS-Rρ, and RvS-RC are all evaluated using the same target reward. Both SDT and RvS-Rρ undergo evaluation with the same target suffix, while CDT, RvS-RC, and other baseline methods are examined using the same target cost of zero. BC-Safe is fed with solely the zero-violation trajectories, but it fails to learn zero-violation policies and exhibits conservative performance and low reward. Our method shows high satisfaction rates, suggesting it consistently adheres to the safety specifications throughout various environments. RvS-Rρ also performs well but cannot realize consistent performance in all environments, e.g. in `Ball-Run` environment, it only has marginal improvement over BC-safe. One reason is that RvS-Rρ does not use sequential modeling as transformers and thus falls short of learning temporal policies. The Q-learning-based algorithms, including BCQ-Lag, BEAR-Lag, and CPQ, as well as COptiDICE, vacillate between excessive conservatism and riskiness. For example, CPQ obtains perfect satisfaction in `Car-Cricle` environment but achieves zero satisfaction in `Ball-Run` and `Drone-Run` environments; BEAR-Lag shows high rewards but also high costs in `Circle` environemnts, suggesting it takes more risks that could lead to unsafe outcomes. However, they have the stitching ability and achieve higher rewards than the conditioned RL baselines whose reward is less than the maximum reward of safe trajectories.

Methods	Ant-Run			Ball-Run			Drone-Run			Average		
	Reward ↑	Cost ↓	Rate ↑	Reward ↑	Cost ↓	Rate ↑	Reward ↑	Cost ↓	Rate ↑	Reward ↑	Cost ↓	Rate ↑
SDT(ours)	**0.95±0.01**	**0.0±0.0**	**1.0±0.0**	0.97±0.01	**0.0±0.0**	**1.0±0.0**	0.97±0.01	0.98±3.43	0.92±0.02	**0.96±0.01**	0.33±2.04	**0.97±0.04**
RvS-Rρ	0.88±0.1	**0.0±0.0**	**1.0±0.0**	0.56±0.37	22.62±26.14	0.5±0.41	0.95±0.06	4.07±16.12	0.87±0.02	0.79±0.28	8.89±20.28	0.79±0.32
CDT	**0.95±0.02**	0.07±0.4	0.97±0.05	**0.98±0.01**	0.63±2.44	0.85±0.18	0.96±0.05	0.25±1.27	0.95±0.04	**0.96±0.03**	**0.32±1.62**	0.92±0.12
RvS-RC	0.87±0.17	0.43±0.99	0.8±0.15	2.02±0.56	49.9±19.22	0.03±0.05	0.91±0.1	10.68±20.09	0.75±0.32	1.27±0.63	20.34±26.69	0.53±0.41
BC-safe	0.92±0.06	0.25±0.77	0.88±0.05	0.52±0.36	9.18±12.72	0.48±0.24	**0.88±0.12**	**0.12±0.78**	**0.97±0.02**	0.78±0.29	3.18±8.5	0.78±0.25
BCQ-Lag	0.8±0.18	2.78±6.03	0.67±0.31	0.83±0.42	10.68±16.04	0.22±0.27	1.06±0.16	76.77±32.92	0.02±0.02	0.9±0.3	30.08±39.49	0.3±0.36
BEAR-Lag	0.01±0.03	**0.0±0.0**	**1.0±0.0**	1.91±1.32	87.67±0.47	0.0±0.0	0.63±0.63	93.23±31.76	0.0±0.0	0.85±1.16	60.3±46.47	0.33±0.47
CPQ	0.03±0.05	**0.0±0.0**	**1.0±0.0**	2.65±1.26	64.33±29.94	0.0±0.0	0.81±0.38	86.53±45.67	0.0±0.0	1.16±1.33	50.29±48.38	0.33±0.47
COptiDICE	0.78±0.07	0.88±3.7	0.9±0.04	0.87±0.04	5.0±4.55	0.33±0.47	1.05±0.08	59.62±10.71	0.0±0.0	0.9±0.13	21.83±27.68	0.41±0.46

Methods	Ball-Circle			Car-Circle			Drone-Circle			Average		
	Reward ↑	Cost ↓	Rate ↑	Reward ↑	Cost ↓	Rate ↑	Reward ↑	Cost ↓	Rate ↑	Reward ↑	Cost ↓	Rate ↑
SDT(ours)	0.86±0.03	**0.48±1.02**	**0.77±0.06**	**0.86±0.02**	2.27±6.15	0.85±0.07	0.94±0.12	**0.07±0.4**	**0.97±0.02**	0.89±0.08	**0.94±3.73**	**0.86±0.1**
RvS-Rρ	0.75±0.06	0.87±1.59	0.68±0.15	0.76±0.09	1.93±4.85	0.82±0.09	0.93±0.03	0.47±1.8	0.9±0.07	0.81±0.11	1.09±3.19	0.8±0.14
CDT	**0.89±0.02**	1.08±1.42	0.5±0.07	0.91±0.01	4.53±6.37	0.42±0.08	**0.95±0.02**	1.07±1.52	0.55±0.04	**0.92±0.03**	2.23±4.2	0.49±0.09
RvS-RC	0.72±0.21	6.93±24.05	0.6±0.15	0.71±0.28	13.42±28.14	0.57±0.02	0.83±0.27	8.6±35.25	0.68±0.05	0.75±0.26	9.65±29.64	0.62±0.1
BC-safe	0.59±0.23	2.22±3.55	0.5±0.15	0.41±0.28	12.45±15.38	0.48±0.14	0.86±0.13	2.58±3.86	0.5±0.11	0.62±0.28	5.75±10.51	0.49±0.13
BCQ-Lag	0.95±0.14	19.78±7.58	0.0±0.0	0.66±0.27	21.8±17.81	0.25±0.11	1.37±0.04	58.38±5.96	0.0±0.0	0.99±0.34	33.32±21.25	0.08±0.13
BEAR-Lag	1.01±0.12	15.35±8.45	0.0±0.0	0.94±0.15	38.8±16.71	0.0±0.0	1.27±0.08	45.45±12.4	0.0±0.0	1.07±0.18	33.2±18.3	0.0±0.0
CPQ	0.79±0.05	2.28±4.01	0.67±0.47	**0.86±0.03**	**0.0±0.0**	**1.0±0.0**	0.12±0.17	14.92±30.51	0.5±0.22	0.59±0.35	5.73±18.94	0.72±0.36
COptiDICE	0.82±0.09	8.28±2.93	0.02±0.02	0.57±0.07	22.95±26.69	0.27±0.12	0.59±0.05	5.98±5.93	0.22±0.06	0.66±0.13	12.41±17.57	0.17±0.13

Methods	Ball-Reach			Car-Reach			Drone-Reach			Average		
	Reward ↑	Cost ↓	Rate ↑	Reward ↑	Cost ↓	Rate ↑	Reward ↑	Cost ↓	Rate ↑	Reward ↑	Cost ↓	Rate ↑
SDT(ours)	**0.61 ± 0.03**	**0.01 ± 0.04**	0.68 ± 0.02	**0.72 ± 0.02**	**-0.01 ± 0.04**	0.62 ± 0.18	**0.91 ± 0.02**	**0.01 ± 0.01**	**0.98 ± 0.03**	**0.75 ± 0.13**	**0.0 ± 0.03**	**0.76 ± 0.19**
RvS-Rρ	0.55 ± 0.07	**0.01 ± 0.03**	**0.7 ± 0.05**	0.3 ± 0.18	**-0.01 ± 0.12**	**0.75 ± 0.2**	0.81 ± 0.14	-0.02 ± 0.12	0.5 ± 0.25	0.55 ± 0.25	-0.01 ± 0.1	0.65 ± 0.22
BC-safe	0.63 ± 0.26	-0.06 ± 0.09	0.22 ± 0.02	0.41 ± 0.29	-0.05 ± 0.19	0.45 ± 0.2	0.69 ± 0.29	-0.01 ± 0.04	0.35 ± 0.05	0.58 ± 0.3	-0.04 ± 0.12	0.34 ± 0.15

Table 4: Complete evaluation results of the normalized reward, cost, and satisfaction rate. ↑: the higher the reward, the better. ↓: the lower the cost (closer to 0), the better. Agents with a higher satisfaction rate than BC-safe are considered safe. Gray: unsafe agents. **Bold**: the best in the respective metric among safe agents.

Target suffix configurations. The different target suffix configurations used in section 5.3 are shown in Figure 6. Due to

the **G** in the specification defined in Eq. (6) and (7), the suffix is monotonically increasing. Recall that positive robustness values indicate that the specification is satisfied and a higher value indicates stronger satisfaction. Therefore, we set positive robustness values as our target suffix for evaluation. As shown in Figure 5, the suffix associated with safe trajectories yielding high rewards is nearly 0, which aligns with the *tempting* concept in (Liu et al., 2022). This suggests that aiming for both a high reward and a high target suffix may lead to reduced performance, as achieving such a combination is impractical. In the environments, a high reward means navigation along the safety boundary, while a high suffix denotes maintaining distance from it. This disparity explains why satisfaction rates of SDT(max) and SDT(mean) decrease compared to SDT(ours). In the case of SDT(linear), a target suffix smaller than that in SDT(ours) is overwhelmed by a high target reward, thus resulting in violations of satisfaction. Empirically, a fixed target suffix works best among these tested configurations.

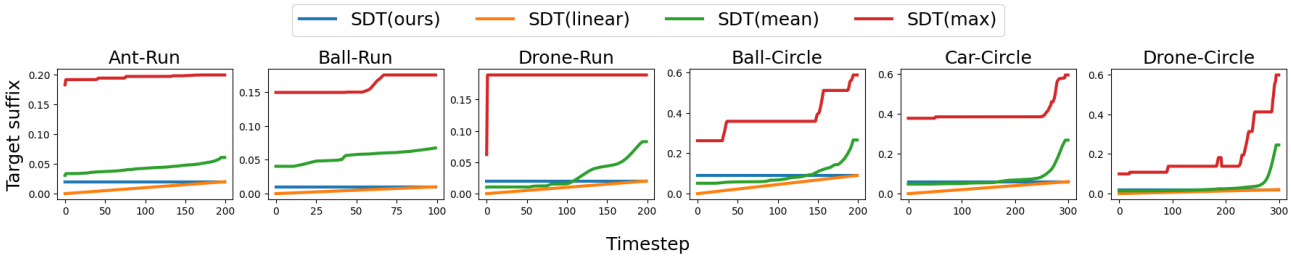

Figure 6: Illustration of different target suffix configurations. i) SDT(ours): fixed target suffix ii) SDT(linear): linearly increasing target suffix; iii) SDT(mean): average suffix from safe trajectories as target suffix; iiii) SDT(max): maximum suffix from safe trajectories as target suffix.

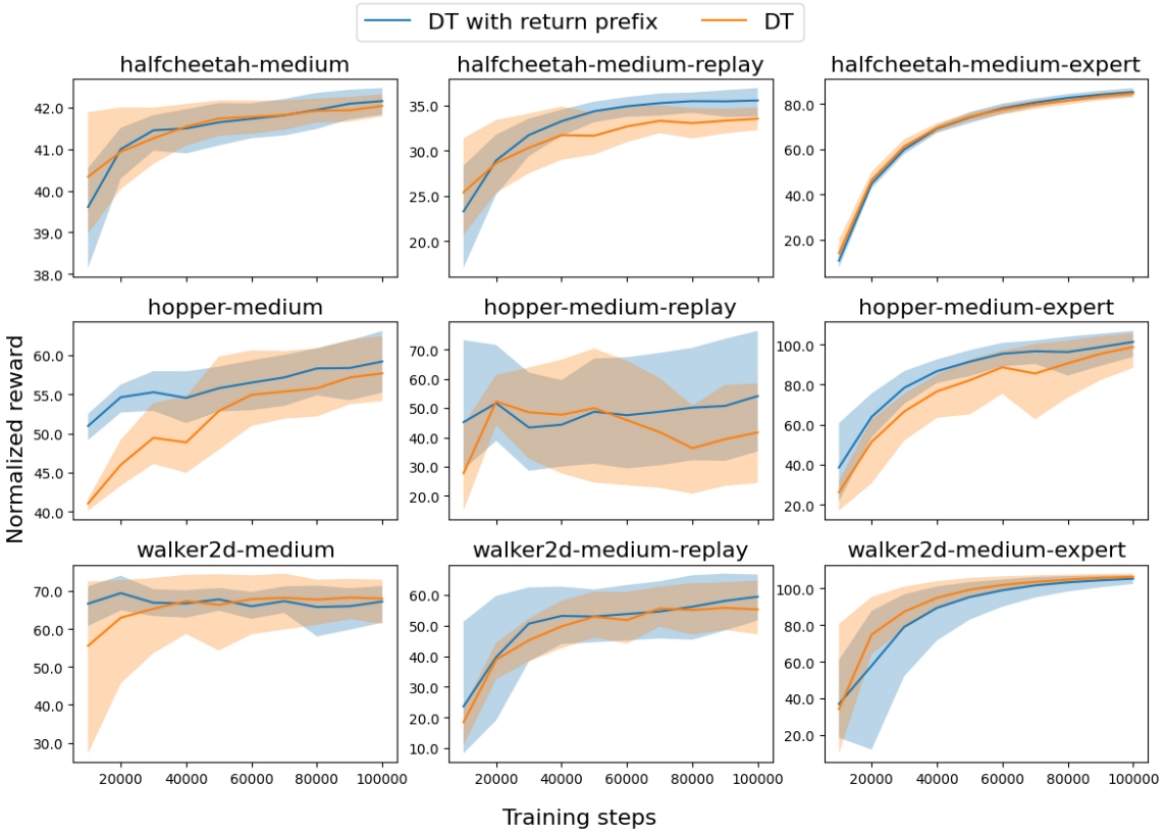

Figure 7: Evaluation results of DT and DT with reward prefix during training. The solid line and the light shade area represent the mean and mean $\pm$ standard deviation. The normalized reward follows the evaluation protocol in `D4RL`.

C. Experiments of Decision Transformers

To demonstrate the role of the prefix, we evaluate the performance of the Decision Transformers (DT) and DT with reward prefix on the `D4RL` benchmark (Fu et al., 2020). We use the official codebase of DT[5] and use the default hyperparameters to train `Halfcheetah`, `Walker2D`, and `Hopper` on different types of dataset: `medium` that collected from a partially-trained policy; `medium-replay` that consists of recording all samples in the replay buffer observed during training until the policy reaches the medium level of performance; and `medium-expert` that contains equal amounts of expert demonstrations and suboptimal data. DT and DT with reward prefix are evaluated every 10000 steps during training and the results are shown in Figure 7. The reward prefix is beneficial not only to the performance after convergence but also to facilitate the training process for most of the tasks since DT with reward prefix achieves higher reward. Another observation is that the impact of the reward prefix is obvious in `medium` and `medium-replay` datasets, while the improvement is marginal in `medium-expert` datasets, indicating that the reward prefix has greater importance in learning good policies from suboptimal data. Empirically, the better performance of introducing the reward prefix supports our augments in section 4.3 that the prefix is not redundant to the suffix but provides additional information that promotes policy learning.

[5]https://github.com/kzl/decision-transformer