Constrained Decoding for Robotics Foundation Models

Anonymous Author(s)
Affiliation
Address
email

Abstract: Recent advances in the development of robotic foundation models have led to promising end-to-end and general-purpose capabilities in robotic systems. These models are pretrained on vast datasets of robot trajectories to process multi-modal inputs and directly output a sequence of action that the system then executes in the real world. Although this approach is attractive from the perspective of improved generalization across diverse tasks, these models are still data-driven and, therefore, lack explicit notions of behavioral correctness and safety constraints. We address these limitations by introducing a constrained decoding framework for robotics foundation models that enforces logical constraints on action trajectories in dynamical systems. Our method ensures that generated actions provably satisfy signal temporal logic (STL) specifications at runtime without retraining, while remaining agnostic of the underlying foundation model. We perform comprehensive evaluation of our approach across state-of-the-art navigation foundation models and we show that our decoding-time interventions are useful not only for filtering unsafe actions but also for conditional action-generation. Videos and code are available on our website: https://constrained-robot-fms.github.io

Keywords: Foundation Models, Provably Safe Robotics, Neuro-symbolic robotics

1 Introduction

Recent advances in Robotics Foundation Models (RFM) have enabled general purpose robot policies that map multi-modal inputs such as RGB images, natural language instructions and proprioceptive inputs to action sequences [1]. RFMs such as SPOC [2], PoliFormer [3], OpenVLA [4] exhibit impressive generalization in navigation and manipulation tasks and serve as versatile robot controllers for real-world deployment contexts. However, these models are primarily data-driven and lack any explicit notion of safety. Although these models may implicitly exhibit safety-related behaviors depending on the patterns in their training data, there is no formal guarantee that models will consistently behave safely in all situations. This serves as a limiting factor for deploying these foundation models in the physical world where rule compliance and regulatory safety rule adherence are crucial.

Formal specifications have long been used to specify rules of operation such as safety requirements or mission directives for robotic deployments [5, 6]. Specifically, *temporal logics* [7] can capture rich desired temporal constraints over robot behavior, such as "remain within permitted region zones and avoid dangerous obstacles". This provides a rigorous workflow to encode behavior rulebooks using temporal logics and enforce them for robotic foundation models. Although temporal logic has seen success in classical robotic planning for reliable behavior generation, its use for foundation models remains limited. Additionally, retraining or fine-tuning these large pre-trained models to directly embed temporal logic specification is challenging [8]. First, retraining models is costly endeavor in terms of computational resources and data requirements. Moreover, due to the stochastic nature of these models, it is difficult to guarantee strict satisfaction of temporal constraints through

Submitted to the 9th Conference on Robot Learning (CoRL 2025). Do not distribute.

training alone. Hence, there is a pressing need for methods that can enforce temporal specifications efficiently at inference time without disrupting the model's pre-trained behavior.

In the field of natural language processing, syntactic constraints have been successfully enforced by applying constrained decoding at inference time [9, 10, 11]. These approaches typically mask out tokens that violate a syntactic constraint defined over token sequences. For example, regular expressions (regex) represent a widely used form of syntactic constraint, requiring that generated token sequences conform to predefined structural patterns [9, 10]. Inspired by this line of work, we extend the paradigm of constrained decoding to enforce logical constraints over action trajectories in dynamical systems and propose *specification aligned decoding (SpecDec)* for RFMs that ensures generated action sequences provably satisfy Signal Temporal Logic (STL) [12] specifications. Our key insight is that decoding-time interventions can be used not just to filter unsafe actions, but to *condition the generation process itself* on specification satisfaction. This conditioning is critical because it steers the model toward generating specification satisfying actions rather than relying on post hoc rejection. SpecDec reduces risk of infeasible outputs while preserving the original action distribution of the model. To enforce such specifications, we leverage the formal semantics of STL to evaluate candidate actions at runtime and mask those that lead to future violations. Our method is agnostic to the underlying foundation model, requiring only two properties: (1) access to the decoding-layer logits during inference, and (2) access to a lightweight dynamics model to predict future states. To efficiently evaluate STL specifications at inference time, we use a high-performance computational graph based library `STLCG++` [13]. To the best of our knowledge, this is the first work to apply constrained decoding with STL specifications to RFMs.

Our main contributions are as follows: First, we introduce the novel problem of *constrained decoding for robotic foundation models* under Signal Temporal Logic specifications (Section 3.1). Second, we propose an inference-time technique that reweights or masks candidate actions using STL satisfaction scores in (Section 3.2 and 3.3). Finally, we demonstrate the effectiveness of our method on state-of-the-art object navigation models without modifying model parameters (Section 4).

2 Preliminaries

2.1 Signal Temporal Logic

Signal Temporal Logic (STL) is an expressive framework for defining properties and reasoning over continuous time real valued signals [14]. Formally, $(\mathbf{s}, t) \models \phi$ denotes that a signal $\mathbf{s}$ satisfies the STL formula ϕ at time t. An atomic predicate of an STL formula is represented by inequalities of the form $\mu(\mathbf{s}(t)) > 0$. The truth value of the predicate μ is equivalent to $\mu(\mathbf{s}(t)) > 0$. Note that with slight abuse of notation, μ represents both the predicate and a function of the trajectory $\mathbf{s}(t)$. Any STL formula consists of Boolean and temporal operations on these predicates, and the syntax of STL formulas is defined recursively as follows:

$$\phi := \mu \mid \neg\mu \mid \phi \wedge \psi \mid \phi \vee \psi \mid \mathbf{G}_{[a,b]}\ \psi \mid \mathbf{F}_{[a,b]}\ \psi \mid \phi\ \mathbf{U}_{[a,b]}\ \psi$$

where ψ and ϕ are STL formulae, $\mathbf{G}$ denotes the globally operator, $\mathbf{F}$ the eventually operator, and $\mathbf{U}$ is the until operator. For example, $\mathbf{s} \models \mathbf{G}_{[a,b]}\psi$ specifies that ψ must be in all times in the given interval, $t \in [a, b]$ of the signal $\mathbf{s}$. Similarly, the operator *until* in $\mathbf{s} \models \phi\mathbf{U}_{[a,b]}\psi$ defines that ϕ must be true until ψ becomes true within a time interval $[a, b]$.

Given a signal s_t representing a signal starting at time t, the Boolean semantics of satisfaction of $s_t \models \phi$ are defined inductively as follows:

$$\begin{aligned}
s_t \models \mu &\iff \mu(s(t)) > 0 \\
s_t \models \neg\varphi &\iff \neg(s_t \models \varphi) \\
s_t \models \varphi_1 \wedge \varphi_2 &\iff (s_t \models \varphi_1) \wedge (s_t \models \varphi_2) \\
s_t \models \mathrm{F}_{[a,b]}(\varphi) &\iff \exists t' \in [t+a, t+b] \text{ s.t. } s_{t'} \models \varphi \\
s_t \models \mathrm{G}_{[a,b]}(\varphi) &\iff \forall t' \in [t+a, t+b] \text{ s.t. } s_{t'} \models \varphi
\end{aligned}$$

Apart from the Boolean semantics, quantitative semantics are defined for a signal to compute a real-valued metric indicating *robustness*, i.e., the strength of satisfaction or violation. For the sake of brevity, the definition of robustness is provided in Appendix A.

2.2 Constrained Decoding in Transformers

A large variety of autoregressive transformer-based models generate final outputs by producing a probability distribution over the model vocabulary at each timestep. This distribution is generated by performing a softmax operation over the model's last hidden layer. Then, through the process of *decoding*, tokens are selected to maximize the overall likelihood of an output sequence. In standard decoding, this maximization can be performed by either greedily selecting the most probable token at each step or by using a beam search to maintain multiple high-likelihood candidates. However, this often leads to degenerate output sequences that are repetitive [15]. A common approach is to use sampling strategies like top k [16], and nucleus sampling [15] that introduce stochasticity to encourage more diverse outputs. Constrained decoding [17] modifies this probabilistic selection by pruning invalid tokens to ensure that the generated sequences satisfy predefined constraints. These constraints are often syntactic, such as regular expressions, JSON formatting, or programming language grammars [18]. There is also recent work on enforcing *semantic* constraints that ensure coherence of the output or alignment with specific knowledge bases [19]. Formally, constrained decoding can be seen as maximizing the probability of the output sequence subject to a constraint $\mathcal{C}$: $\arg\max_{y \in \mathcal{Y}_\mathcal{C}} P(y \mid x)$ where $\mathcal{Y}_\mathcal{C}$ is the set of sequences satisfying $\mathcal{C}$.

3 Specification-Guided Constrained Decoding

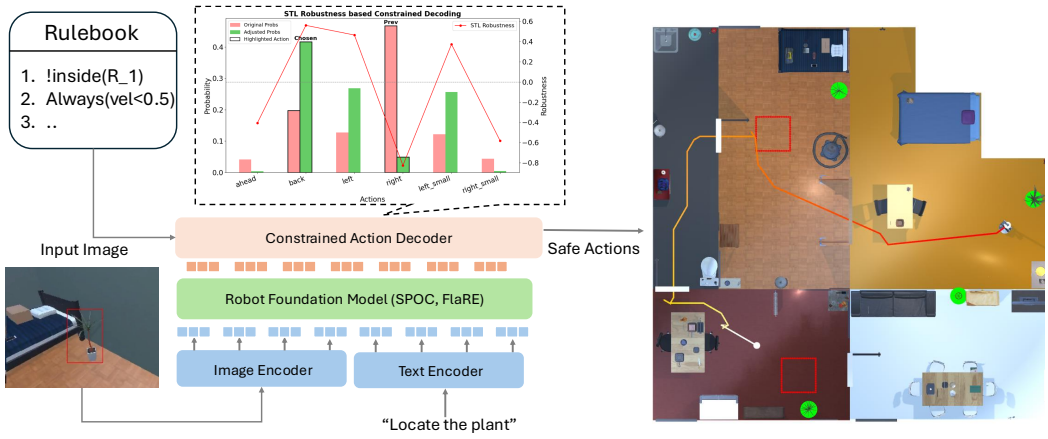

Figure 1: Overview of our specification aligned decoding framework. Given multimodal inputs (e.g., RGB images and natural language instructions), a pretrained Robot Foundation Model (e.g., SPOC) generates candidate actions. These actions are filtered or reweighted by the constrained decoding technique based on temporal logic constraints. In the figure, green markers denote target object locations, while red zones represent regions to avoid, as defined by temporal logic constraints.

In this section, we introduce a novel problem formulation for SpecDec in RFMs. First, we highlight the challenge in specification checking for RFMs in contrast to traditional syntactical constraint checking adopted by LLMs, and our solution to remedy it. Then, we propose two novel inference-time techniques for specification aligned decoding: *Hard Constrained Decoding* (HCD) and *Robustness Constrained Decoding* (RCD).

3.1 Problem Statement

As highlighted in the background section, existing techniques in constrained decoding for language models enforce *syntactic constraints* defined over tokens such as conforming to a context-free grammar or matching a regular expression. In these setups, constraint checking can be performed in the model's token space.

In contrast, RFMs operate in a physical environment and constraints (captured via temporal logic) are defined over state variables. Since a large class of end-to-end RFMs solely propose action sequences, specification checking can only be performed as actions are executed and the environment is simulated forward. In this case, constrain checking cannot be done solely in the token space and requires environmental feedback or a dynamics stepping function. Hence, we leverage a dynamics function to compute specification satisfaction of different action sequences proposed by an RFM.

Consider a discrete dynamical system with states $x_t \in \mathbb{R}^n$ and actions $a_t \in \mathbb{A}$ at time step t. The system's dynamics are defined by $x_{t+1} = f(x_t, a_t)$ where $f : \mathbb{R}^n \times \mathbb{A} \to \mathbb{R}^n$ maps the current state $(x_t \in \mathbb{R}^n)$ and a discrete action $(a_t \in A)$ to the next state $x_{t+1} \in \mathbb{R}^n$. This system is controlled by a policy that selects an action a_t at each time step based on observations and task context such as user-provided natural-language instructions or goal waypoints.

In this work, we focus on RFMs that generate actions based on multi-modal inputs, including sensor observations (e.g., RGB, depth, LiDAR) and natural language instructions. Let $\mathcal{I}_t$ represent the aggregated input at timestep t. These inputs are first encoded into a latent embedding space through modality-specific encoders: $e_{\mathcal{I}_t} = \mathcal{E}_{\mathcal{I}}\big(\mathcal{I}_t\big)$.

Given the history of encoded inputs up to time t, a Transformer-based foundation model parameterized by θ predicts embeddings for the next $T-t$ actions:

$$\{\hat{e}_{a_{t+k}}\}_{k=1}^{T-t} = \text{Transformer}_\theta\big(\{e_{\mathcal{I}_\tau}\}_{\tau=0}^{t}\big).$$

Each predicted action embedding is decoded into an action $\hat{a}_{t+k} \in \mathbb{A}$, resulting in a predicted action sequence $\{\hat{a}_{t+1}, \ldots, \hat{a}_T\}$, where T denotes the planning horizon.

Now, consider that the system is required to satisfy requirements encoded using an STL formula φ defined over the state variables of the system. Formally, the goal is to ensure that the resulting trajectory satisfies the specification φ:

$$\{(x_0, \hat{a}_0), \ldots, (x_T, \hat{a}_T)\} \models \varphi$$

Most of the existing techniques for specification enforcement perform posthoc manipulation of proposed foundation model actions through filtering or rejecting action sequences that violate the specification φ. Although manipulation after sampling can ensure specification satisfaction, it can lead to distorting the model's learned distribution, producing low likelihood outputs. This undermines the inductive biases learned during pretraining and leads to degenerate, brittle behaviors. A similar problem was highlighted when ensuring compliance with logical constraints for large language models in [20]. Additionally, RFMs decode actions sequentially, where each action a_t is conditioned on previously generated tokens $a_{<t}$. Posthoc manipulation can disrupt this causal chain and lead to a mismatch between the model's internal hidden state and the executed sequence. Hence, we propose the following problem statement:

> *How can we enforce temporal logic constraints during action generation in robot foundation models such that the output sequence (1) satisfies an STL specification φ, and (2) remains faithful to the model's autoregressive distribution $\pi(a_{1:T} \mid \mathcal{I}_{1:T})$?*

Let $\pi(a_{1:T})$ be the *unconstrained* action-sequence distribution produced by the RFM's decoder (e.g. the softmax over logits generated by the Transformer). We define the ideal constrained distribution over action sequences as:

$$Q_{\pi,\varphi}(a_{1:T}) = \frac{\pi(a_{1:T}) \cdot \mathbf{1}[(x_{1:T}, a_{1:T}) \models \varphi]}{\sum_{a'_{1:T}} \pi(a'_{1:T}) \cdot \mathbf{1}[(x'_{1:T}, a'_{1:T}) \models \varphi]} \tag{1}$$

where $x_{1:T}$ denotes the state trajectory induced by the system dynamics under actions $a_{1:T}$ and $\mathbf{1}[\cdot]$ is the indicator function that returns 1 iff the trajectory-action pair satisfies the specification. Equation 1 is the exact Bayesian conditioning of π on the event that the generated rollout satisfies φ. Hence, sampling from $Q_{\pi,\varphi}$ would give sequences that (i) inherit the original model's preferences encoded in π and (ii) *guarantee* specification satisfaction.

In this work, we propose a technique to overcome the drawbacks of post-hoc safety enforcement methods (such as filtering) by leveraging constrained decoding techniques. Specifically, we propose SpecDec: A constrained decoding strategy that integrates STL specifications into the foundation model action selection process itself, ensuring satisfaction without distorting the model's distribution.

3.2 Hard Constrained Decoding

As highlighted in the background section, in the final layer, predictions are detokenized and a projection layer converts the embeddings into logits over the vocabulary space. These logits are further converted into a probability distribution using a softmax operation. In prior work, for structured output generation in LLMs, some invalid tokens are masked based on syntactical constraints or other criteria [18, 20]. This is done by setting their logit value as $-\infty$ before the softmax operation is applied. For HCD, we use a similar approach as constrained decoding literature [18] and mask out predicted action tokens that violate our given STL specification φ during sequential generation. Formally, to enforce the STL specification φ during sequential generation, we adjust the logits at each timestep $t+k$ as follows:

Let $\mathbf{z}_{t+k}$ denote the logits at timestep $t+k$. For each action choice i at timestep $t+k$, we define:

$$z_{t+k}^{(i)} = \begin{cases} -\infty, & \text{if } \hat{x}_{t+k}^{(i)} = f(x_{t+k-1}, \hat{a}_{t+k}^{(i)}) \text{ violates } \varphi \\ z_{t+k}^{(i)}, & \text{otherwise} \end{cases}$$

Here t is the current decision step, k is an index for the look-ahead step $t+k$ within a planning horizon of length T ($k \in [1..T]$), $\hat{a}_{t+k}^{(i)}$ is the action mapping to the token i and $\hat{x}_{t+k}^{(i)}$ is the next state value upon taking this action. This next state is elicited using a simple dynamics model (f) as highlighted in the previous section. Adjusting logits in this fashion ensures that any invalid token with respect to the safety specification will have zero probability of being selected after applying the softmax function. We introduce a theorem to state that our proposed approach generates a trajectory that satisfies the given specification φ:

Theorem 1 (Specification Satisfaction Modulo Dynamics)**.** *Let π_{HCD} denote the policy induced by hard-constrained decoding with respect to an STL specification φ, and let f be a deterministic dynamics model used during decoding. Then the generated trajectory satisfies the specification:*

$$\{(x_{t+1}, \hat{a}_{t+1}), \ldots, (x_T, \hat{a}_T)\} \models \varphi$$

where each action $\hat{a}_{t+k} \sim \pi_{HCD}(\cdot \mid x_{t+k-1})$, and $x_{t+k} = f(x_{t+k-1}, \hat{a}_{t+k})$ for all k.

A proof sketch is provided in the Appendix B

3.3 Robustness Constrained Decoding

HCD ensures compliance but can lead to compromising task success, which can be undesirable. A similar tradeoff was observed by [21] when probability space-steering preserved model fluency while reducing toxic continuations compared with hard-filtering strategies that inflated perplexity and eroded diversity. Hence, we propose an alternative approach, called RCD, where we leverage the quantitative semantics of STL specifications (robustness). Unlike HCD, which applies hard masking to completely remove unsafe actions, RCD softly guides the model toward safer actions by incorporating robustness scores that reflect the degree of satisfaction of φ. This is similar to the approach proposed in [21] where the next-token distributions were re-weighted based on the utility scores provided by another language model. Our utility scores are quantified by the robustness

function($\rho(x_t, \varphi)$) that returns a real-valued score indicating how well a predicted state satisfies the specification. Positive robustness values denote specification satisfaction, while negative values capture the degree of violation.

First, we compute a robustness score for each candidate action: $r_{t+k}^{(i)} = \rho(\hat{x}_{t+k}^{(i)}, \varphi)$ where $\rho(\cdot, \varphi)$ is the STL robustness metric, and $\hat{x}_{t+k}^{(i)}$ is the predicted next state under action $\hat{a}_{t+k}^{(i)}$. This robustness score $r_{t+k}^{(i)}$ quantifies how well each candidate action satisfies the specification φ. These scores are then converted into weights using exponential scaling: $w_{t+k,i} = \exp(\alpha \cdot r_{t+k,i})$ where α is a temperature parameter that adjusts the sharpness of the bias. We use these weights to shift the original logits: $\tilde{z}_{t+k,i} = z_{t+k,i} + \beta \cdot w_{t+k,i}$ where β is a hyperparameter that modulates the trade-off between specification adherence and the original task objective. Finally, we obtain the action distribution by applying softmax over the adjusted logits: $p_{t+k} = \text{softmax}(\tilde{\mathbf{z}}_{t+k})$

This approach allows for graded preferences that improve flexibility and robustness to dynamics approximation errors. This is because every action keeps a non-zero probability, the policy can still recover when the dynamics model is imperfect. Concretely, if the predicted successor $\hat{x}_{t+1}$ is off by ϵ, an action that looked marginally unsafe can be safe in the true system, and vice versa. Retaining a weighted down probability for this action gives the sampler a fall-back option where as HCD would completely rule this action out due to 0 probability. Since we are shifting the probability mass for unsatisfying actions, it is possible that they are still chosen and lead to a violation. However, this is a tradeoff we allow to achieve a given task objective. We note that this still ensures higher STL sastifaction than unconstrained actions.

4 Evaluation

4.1 Implementational Details

We evaluate our constrained decoding approach on AI2-THOR simulated [22] indoor environments using the state-of-the-art (SOTA) navigation model called Shortest Path Oracle Clone (SPOC) [2]. This model achieves high navigation success rates for diverse tasks such as object navigation, room navigation, and so forth. We encode the STL specifications using an efficient computational graph-based STL library called STLCG++ that can evaluate multiple state signals in parallel [13]. This ensures minimal inference overhead at runtime, which is crucial for foundation model deployment. We enforce geofencing and obstacle avoidance by encoding them into invariant STL safety specifications. Specifically, we generate random regions in the configuration space that the robot must either avoid (obstacle zones) or remain within (safe zones), and apply these constraints in real time during execution.

4.2 Benchmarks and Research Questions

We compare our proposed techniques with (1) an unconstrained base model and (2) a base model with a filtering mechanism. The filtering mechanism picks a default action (turning left or right in place) upon predicted violation of the safety specification, similar to the Simplex architecture [23]. Simplex architecture is a classic scheme in which a high-performance advanced controller is continuously monitored by a provably safe but less capable backup controller. Simplex based techniques have been used extensively for safety-critical robotics and are a widely accepted standard for runtime-safety comparisons. We are interested in three main metrics: **STL Satisfaction Rate** (STL St): Proportion of trajectories that satisfy the specified STL formula, **Task Success rate** (SR): Standard task success, **Average Robustness Score** (AR): A quantitative measure of how strongly STL constraints are satisfied. The three main research questions we investigated in this paper:

1. **RQ1**: Do RCD and HCD outperform the baselines in STL satisfaction?
2. **RQ2**: Do RCD and HCD lead to task success rates comparable to those of the unconstrained technique?

3. **RQ3**: Does RCD perform better than HCD for task success rates?

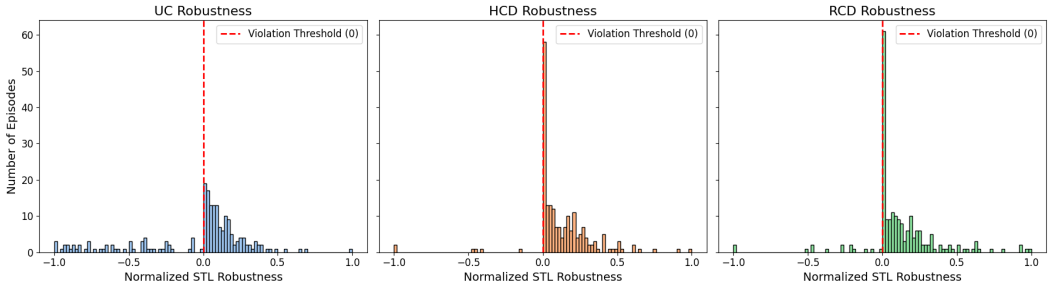

Figure 2: Distribution of STL robustness scores across three action selection techniques for ϕ_{avoid}. The red dashed line marks the violation threshold at robustness = 0. HCD achieves the lowest violation rate, while RCD strikes a balance between task success and STL satisfaction.

Specification	Decoding	STL St (%)	SR (%)	AR
ϕ_{avoid}	Unconstrained (SPOC)	72.00	82.5	1.60
	Filtering (SPOC-F)	97.00	72.00	1.57
	Hard-Constrained (SPOC-HCD)	97.00	72.50	1.73
	Robust-Constrained (SPOC-RCD)	93.00	76.00	1.71
$\phi_{geofence}$	Unconstrained (SPOC)	78.00	81.5	1.68
	Filtering (SPOC-F)	100.00	72.00	1.89
	Hard-Constrained (SPOC-HCD)	100.00	76.5	1.97
	Robust-Constrained (SPOC-RCD)	95.35	80.23	1.86

Table 1: Comparison of decoding strategies across different backbone models on STL-constrained tasks. Our constrained decoding approaches show significant improvements in STL satisfaction and robustness while preserving task success.

4.3 Results

Our results are highlighted in Table 1 and Figure 2. Unless stated otherwise, all numbers are averaged over 200 evaluation episodes.

RQ1 – STL satisfaction. Both HCD and RCD improve STL satisfaction compared with the unconstrained SPOC controller. For ϕ_{avoid}, HCD improves satisfaction from $72\,\%$ to $97\,\%$, and RCD reaches $93\,\%$. This trend is even stronger for $\phi_{geofencing}$: HCD achieves perfect compliance ($100\,\%$), RCD attains $95.4\,\%$, whereas the base model satisfies only $78\,\%$. We observe that the Simplex-style filtering baseline achieves similar STL-satisfaction rate as HCD, $97\,\%$ for ϕ_{avoid} and $100\,\%$ $\phi_{geofence}$. This parity is expected as both methods block any action predicted to violate the specification.

RQ2 – Task completion. Simplex-style filtering sacrifices task success (SR) because the agent takes predefined safe actions. HCD achieves high STL satisfaction, but solves 6-9$\%$ fewer tasks succesfully than the unconstrained controller. However, as HCD factors in base model logits, it is able to achieve higher task satisfaction for $\phi_{geofence}$ compared to Simplex-style filtering. By contrast, RCD preserves task success almost at the level of the unconstrained policy: $76\,\%$ versus $82\,\%$ for the ϕ_{avoid} and $80.2\,\%$ versus $81.5\,\%$ for $\phi_{geofencing}$. This is significantly higher than Simplex-style filtering on the same task.

RQ3 – RCD vs. HCD. While both HCD and RCD improve safety over the unconstrained baseline, they differ in how they balance constraint satisfaction with task success. HCD enforces strict

STL satisfaction that results in frequent conservatism and lower successful task completion rates. In contrast, RCD's soft penalization leads to higher task success—up to 4–5 % more than HCD—while still maintaining over 93 % STL satisfaction. These results show that RCD achieves a better trade-off between safety and goal-directed behavior, especially in settings where occasional low-risk actions can lead to higher long-term rewards. We also perform an ablation for varying values of β to investigate this tradeoff and provide these results in Appendix C.

Our proposed techniques effectively enforces STL specifications during policy execution. HCD ensures full compliance but occasionally sacrifices task success due to strict truncation. RCD strikes a balance, offering high satisfaction rates and robust performance. This highlights the feasibility of combining learning-based models with formal safety constraints.

5 Related Work

Constraint satisfaction for robotics has been an active area of research that involves techniques such as control barrier functions (CBFs) [24], safe reinforcement learning [25], and temporal logic-based shielding approaches [26]. Recently, with the advent of Vision Language Action models and their impressive generalizable capabilities for manipulation, navigation and other tasks, there are growing concerns about ensuring safety and correctness without retraining these large models. Although classical methods offer formal guarantees, they require pretraining/fine-tuning stage interventions, which can be restrictive. For example, SafeVLA [27] fine-tunes pre-trained foundation models with task-specific safety costs, achieving strong performance in Safety-CHORES tasks. However, the safety specification is expected to be embedded in the training data and loss, meaning the model cannot generalize to new safety constraints at test time. In contrast, ASIMOV [28] explores rewriting dangerous instructions with better human-aligned alternatives to steer model behavior without modifying model parameters, but lacks trajectory-level formal guarantees. Our technique achieves a middle ground with the ability to adapt to novel specifications at test time without modifying model parameters while requiring minimal assumptions about the underlying model and incurring negligible computational overhead. The closest to our work is SELP [29] that proposes LTL-constrained decoding for language model-based plan generation. However, since LTL does not possess quantitative semantics, they are restricted to hard masking-based approach.

6 Limitations and Future Work

In this work, we introduce a constrained decoding framework for enforcing STL specifications for robotic foundation models. Our approach enables runtime adaptation to novel safety specifications without retraining. Through experiments across multiple simulated environments, we demonstrated that our method significantly improves STL satisfaction while maintaining high task success rates. Our approach makes two critical assumptions that can be a limiting factor. First, we assume access to specifications that are defined over the state space and that these specifications are generated by roboticists. Although this is a common situation for safety critical deployment contexts like aerial robotics [30, 31], these specifications can be difficult to design and involve access to a localization module that can provide accurate state estimation. We hope to remedy this bottleneck by leveraging open-world safety specifications using recent work on embedding spaces-based logic (ETL) [32] and using Large Language Models for generating high level specifications automatically [33]. Second, our approach also assumes access to a predictive model to evaluate the impact of actions on future trajectories. This can limit applicability to settings where accurate dynamics models are unavailable. However, it is possible to mitigate this via learned dynamics models [34] or world models proposed in [35, 36]. In addition, while our experiments are conducted in simulation, the underlying foundation model (SPOC) has been successfully deployed on real robots with minimal sim-to-real degradation [2]. Since our constrained decoding operates primarily at inference time and leaves the base model unchanged, the observed improvements in STL satisfaction and task success are expected to transfer over to the real world. We plan to rigorously test this by deploying SpecDec on the same physical robot platform as [2].

References

[1] Y. Hu, Q. Xie, V. Jain, J. Francis, J. Patrikar, N. V. Keetha, S. Kim, Y. Xie, T. Zhang, S. Zhao, et al. Toward general-purpose robots via foundation models: A survey and meta-analysis. *CoRR*, 2023.

[2] K. Ehsani, T. Gupta, R. Hendrix, J. Salvador, L. Weihs, K.-H. Zeng, K. P. Singh, Y. Kim, W. Han, A. Herrasti, R. Krishna, D. Schwenk, E. VanderBilt, and A. Kembhavi. Spoc: Imitating shortest paths in simulation enables effective navigation and manipulation in the real world, 2024. URL https://arxiv.org/abs/2312.02976.

[3] K.-H. Zeng, Z. Zhang, K. Ehsani, R. Hendrix, J. Salvador, A. Herrasti, R. Girshick, A. Kembhavi, and L. Weihs. Poliformer: Scaling on-policy rl with transformers results in masterful navigators, 2024. URL https://arxiv.org/abs/2406.20083.

[4] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. Openvla: An open-source vision-language-action model, 2024. URL https://arxiv.org/abs/2406.09246.

[5] C. Menghi, C. Tsigkanos, P. Pelliccione, C. Ghezzi, and T. Berger. Specification patterns for robotic missions. *CoRR*, abs/1901.02077, 2019. URL http://arxiv.org/abs/1901.02077.

[6] M. Farrell, M. Luckcuck, and M. Fisher. *Robotics and Integrated Formal Methods: Necessity Meets Opportunity*, page 161–171. Springer International Publishing, 2018. ISBN 9783319989389. doi:10.1007/978-3-319-98938-9_10. URL http://dx.doi.org/10.1007/978-3-319-98938-9_10.

[7] A. Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57, 1977. doi:10.1109/SFCS.1977.32.

[8] P. Kapoor, S. Vemprala, and A. Kapoor. Logically constrained robotics transformers for enhanced perception-action planning. *arXiv preprint arXiv:2408.05336*, 2024.

[9] B. T. Willard and R. Louf. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*, 2023.

[10] L. Beurer-Kellner, M. Fischer, and M. Vechev. Prompting is programming: A query language for large language models. *Proc. ACM Program. Lang.*, 7(PLDI), June 2023. doi:10.1145/3591300. URL https://doi.org/10.1145/3591300.

[11] G. AI. Guidance: A language model control framework. https://github.com/guidance-ai/guidance, 2023. Accessed: April 27, 2025.

[12] O. Maler and D. Nickovic. Monitoring Temporal Properties of Continuous Signals. (3253): 152–166, 2004.

[13] P. Kapoor, K. Mizuta, E. Kang, and K. Leung. Stlcg++: A masking approach for differentiable signal temporal logic specification, 2025. URL https://arxiv.org/abs/2501.04194.

[14] O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *FORMATS/FTRTFT*, 2004. URL https://api.semanticscholar.org/CorpusID:15642684.

[15] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

[16] A. Fan, M. Lewis, and Y. Dauphin. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*, 2018.

[17] C. Hokamp and Q. Liu. Lexically constrained decoding for sequence generation using grid beam search. In R. Barzilay and M.-Y. Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi:10.18653/v1/P17-1141. URL https://aclanthology.org/P17-1141/.

[18] S. Welleck, A. Bertsch, M. Finlayson, H. Schoelkopf, A. Xie, G. Neubig, I. Kulikov, and Z. Harchaoui. From decoding to meta-generation: Inference-time algorithms for large language models, 2024. URL https://arxiv.org/abs/2406.16838.

[19] M. Peyrard, M. Josifoski, and R. West. The era of semantic decoding, 2024. URL https://arxiv.org/abs/2403.14562.

[20] K. Park, J. Wang, T. Berg-Kirkpatrick, N. Polikarpova, and L. D' Antoni. Grammar-aligned decoding. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 24547–24568. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/2bdc2267c3d7d01523e2e17ac0a754f3-Paper-Conference.pdf.

[21] A. Liu, M. Sap, X. Lu, S. Swayamdipta, C. Bhagavatula, N. A. Smith, and Y. Choi. Dexperts: Decoding-time controlled text generation with experts and anti-experts, 2021. URL https://arxiv.org/abs/2105.03023.

[22] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, M. Deitke, K. Ehsani, D. Gordon, Y. Zhu, A. Kembhavi, A. Gupta, and A. Farhadi. Ai2-thor: An interactive 3d environment for visual ai, 2022. URL https://arxiv.org/abs/1712.05474.

[23] L. Sha. Using simplicity to control complexity. *IEEE Software*, 18(4):20–28, 2001. doi:10.1109/MS.2001.936213.

[24] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada. Control barrier functions: Theory and applications, 2019. URL https://arxiv.org/abs/1903.11199.

[25] S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, and A. Knoll. A review of safe reinforcement learning: Methods, theory and applications, 2024. URL https://arxiv.org/abs/2205.10330.

[26] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu. Safe reinforcement learning via shielding, 2017. URL https://arxiv.org/abs/1708.08611.

[27] B. Zhang, Y. Zhang, J. Ji, Y. Lei, J. Dai, Y. Chen, and Y. Yang. Safevla: Towards safety alignment of vision-language-action model via safe reinforcement learning, 2025. URL https://arxiv.org/abs/2503.03480.

[28] P. Sermanet, A. Majumdar, A. Irpan, D. Kalashnikov, and V. Sindhwani. Generating robot constitutions & benchmarks for semantic safety. *arXiv preprint arXiv:2503.08663*, 2025.

[29] Y. Wu, Z. Xiong, Y. Hu, S. S. Iyengar, N. Jiang, A. Bera, L. Tan, and S. Jagannathan. Selp: Generating safe and efficient task plans for robot agents with large language models, 2025. URL https://arxiv.org/abs/2409.19471.

[30] J. J. Aloor, J. Patrikar, P. Kapoor, J. Oh, and S. Scherer. Follow the rules: Online signal temporal logic tree search for guided imitation learning in stochastic domains. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1320–1326, 2023. doi:10.1109/ICRA48891.2023.10160953.

[31] M. Luckcuck, M. Farrell, L. A. Dennis, C. Dixon, and M. Fisher. Formal specification and verification of autonomous robotic systems: A survey. *ACM Computing Surveys*, 52(5):1–41, Sept. 2019. ISSN 1557-7341. doi:10.1145/3342355. URL http://dx.doi.org/10.1145/3342355.

[32] P. Kapoor, A. Hammer, A. Kapoor, K. Leung, and E. Kang. Pretrained embeddings as a behavior specification mechanism, 2025. URL https://arxiv.org/abs/2503.02012.

[33] M. Li, S. Zhao, Q. Wang, K. Wang, Y. Zhou, S. Srivastava, C. Gokmen, T. Lee, L. E. Li, R. Zhang, W. Liu, P. Liang, L. Fei-Fei, J. Mao, and J. Wu. Embodied agent interface: Benchmarking llms for embodied decision making, 2025. URL https://arxiv.org/abs/2410.07166.

[34] M. Lutter, L. Hasenclever, A. Byravan, G. Dulac-Arnold, P. Trochim, N. Heess, J. Merel, and Y. Tassa. Learning dynamics models for model predictive agents, 2021. URL https://arxiv.org/abs/2109.14311.

[35] G. Zhou, H. Pan, Y. LeCun, and L. Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning, 2025. URL https://arxiv.org/abs/2411.04983.

[36] V. Micheli, E. Alonso, and F. Fleuret. Transformers are sample-efficient world models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=vhFu1Acb0xb.

Appendix

A Quantiative Semantics of STL

Given a signal s_t representing a signal starting at time t, the quantitative semantics of satisfaction of $s_t \models \phi$ are defined inductively as follows:

$$\begin{aligned}
\rho(s_t, \mu_c) &= \mu(x_t) - c \\
\rho(s_t, \neg\varphi) &= -\rho(s_t, \varphi) \\
\rho(s_t, \varphi_1 \wedge \varphi_2) &= \min(\rho(s_t, \varphi_1), \rho(s_t, \varphi_2)) \\
\rho(s_t, \mathrm{F}_{[a,b]}(\varphi)) &= \max_{t' \in [t+a, t+b]} \rho(s'_t, \varphi) \\
\rho(s_t, \mathrm{G}_{[a,b]}(\varphi)) &= \min_{t' \in [t+a, t+b]} \rho(s'_t, \varphi)
\end{aligned}$$

B HCD proof

We rewrite the theorem here for easier comprehension.

Let π_{HCD} denote the policy induced by hard-constrained decoding with respect to an STL specification φ, and let f be a deterministic dynamics model used during decoding. Then the generated trajectory satisfies the specification:

$$\{(x_{t+1}, \hat{a}_{t+1}), \ldots, (x_T, \hat{a}_T)\} \models \varphi$$

where each action $\hat{a}_{t+k} \sim \pi_{\text{HCD}}(\cdot \mid x_{t+k-1})$, and $x_{t+k} = f(x_{t+k-1}, \hat{a}_{t+k})$ for all k.

Proof Sketch.

$$\frac{\pi_{\text{HCD}}(\hat{a}_{t+k} \mid x_{t+k-1}) = \{0 \mid \forall \hat{a}^{(i)}_{t+k} \in \mathbb{A}, \hat{x}^{(i)}_{t+k} \not\models \varphi \wedge \hat{x}^{(i)}_{t+k} = f(\hat{x}_{t+k-1}, \hat{a}^{(i)}_{t+k}\}}{\{(x_{t+1}, a_{t+1}), \ldots, (x_T, a_T)\} \models \varphi} \text{HCD-SAFE}$$

We proceed by induction over time. At step $t+1$, the masking ensures that the selected action a_{t+1} is such that $(x_{t+1}, a_{t+1}) \models \varphi$, where $x_{t+1} = f(x_t, a_{t+1})$. Assuming that the sequence $\{(x_{t+1}, a_{t+1}), \ldots, (x_{t+k-1}, a_{t+k-1})\}$ satisfies φ, we show that (x_{t+k}, a_{t+k}) will also satisfy φ. Since actions are masked based on predicted rollouts using f, and f is exact, the only permitted actions at step $t+k$ are those that extend the sequence without violating φ. By induction, the entire sequence satisfies φ. □

C Robustness weighting ablation

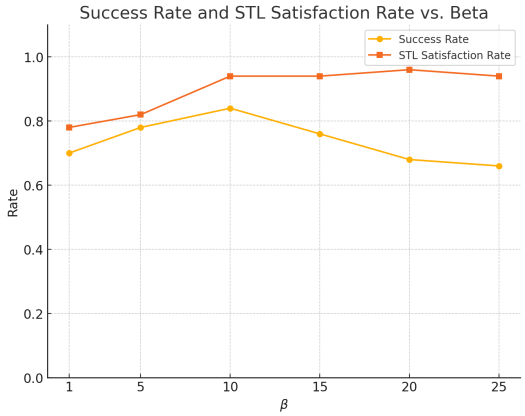

Figure 3: Success rate and STL satisfaction rate across different values of β for RCD, which controls the importance of STL spcification during decoding. As β increases, STL satisfaction improves while task success shows a slight tradeoff beyond moderate values.

To evaluate the impact of relative weighting between robustness and base logits, we plotted the success rate and STL satisfaction rate across varying values for β. As shown in Figure 3, increasing the robustness threshold initially improves both task success and specification satisfaction, with success rates peaking around a moderate threshold (5–10). However, at higher thresholds (20 and 25), success rates begin to decline slightly, while STL satisfaction remains high.

This trend highlights a trade-off introduced by weighting robustness. Higher robustness weighting leads to better STL satisfaction at the cost of hindering overall task completion as logits for feasible actions are shifted heavily by the robustness values. These results demonstrate the importance of carefully balancing robustness weighting to maintain both task success and STL satisfaction.