Formal Methods for Learning-Enabled Safe Autonomy

Author Names Omitted for Anonymous Review

I. Introduction

Foundation models and world models are turning perception, language understanding, and planning into reusable capabilities, enabling many robot form factors to operate across diverse environments with unprecedented flexibility [11]. Yet deploying these models safely remains an open problem. Safety-critical autonomy requires strict behavioral constraints, such as collision avoidance, geofencing, sequencing rules, and temporal deadlines that must be satisfied even under distribution shift [16]. Today's data-driven policies can produce capable actions, but they lack explicit notions of correctness: they provide no guarantees of constraint satisfaction, and their failures are difficult to diagnose and systematically repair. In open-world settings, this gap is a core barrier to trustworthy deployment [8].

Formal methods provide the missing layer for safety-critical autonomy: a mathematically grounded way to specify what a robot must do (and must never do), and to reason about those requirements with guarantees. My research takes lessons from reliable software engineering lifecycle and applies them to learning-enabled robots, where temporal logic [17] specifications such as signal temporal logic (STL) [15] ensure "correctness" at three stages for general aviation, mobile manipulation, and navigation regimes. My first thrust focuses on making fundamental advances in imitation learning, robotics transformers and temporal logic specification decomposition so trajectories are constructed to satisfy requirements by construction [19, 20, 2]. My second thrust focuses on analysis where specifications provide a rigorous lens for iterative improvement: they enable systematic falsification for analyzing policies under distribution shift, classifying generated failures into actionable diagnoses via a novel counterexample classification mechanism, and automated repair mechanisms for both specifications and policies [21, 24, 4] . Finally, in enforcement, specifications serve as the last line of defense when reality departs from assumptions: runtime guards monitor execution under uncertainty and distribution shift and trigger intervention (e.g., constraining action generation using constrained decoding or invoking safety filters) before the system enters states where violation becomes inevitable [3, 23, 1]. To enable all three thrusts, I also developed a scalable differentiable robustness engine using computational graphs [22], which makes specifications usable inside modern learning and evaluation loops. Overall, my research leverages formal methods to ensure learning-driven autonomy satisfies strict behavioral requirements, enabling generalist models to be deployed in domains where mistakes are unacceptable.

II. Past and Current Work

A. Synthesis: correct-by-construction behavior design

The most reliable path to safe autonomy is to enforce correctness at design time: instead of checking specifications after the fact, I use temporal-logic semantics to directly synthesize behaviors that satisfy the requirements. However, classical temporal logic is not a drop-in tool for modern data-driven policies that can guide imitation, scale to long horizons, or integrate with transformer-based trajectory generation [16]. My synthesis work develops the missing machinery to make temporal logic usable in modern pipelines. First, we enhance learning-from-demonstration [18] policies with rule compliance by introducing an online tree search algorithm that uses demonstrations to propose likely actions while using temporal-logic robustness to certify rule satisfaction. Instead of treating LfD as pure imitation, the search treats the demonstration as a prior over the action space and then performs lookahead to select actions that remain on a satisfying continuation of the specification [19]. This creates a principled bridge between data-driven imitation and formal constraints. To make this approach scale to long-horizon specifications with deep temporal nesting, I develop a sound incremental decomposition technique that breaks complex STL specifications into tractable subproblems whose solutions compose, avoiding combinatorial blowups [13] while preserving end-to-end correctness guarantees [20]. Finally, to make correct-by-construction compatible with foundation-model policy stacks, I integrated logical specifications into robotics transformers [7] by encoding the STL formula into spec tokens and using cross-attention to condition perception–action trajectory generation on those tokens at each decoding step, so the specification shapes the model's trajectory distribution directly

[2]. Together, these contributions combine modern data driven policies with temporal logic specifications.

B. Analysis: specification repair, falsification, and counterexample understanding

Specifications are only useful in practice if they support an *engineering loop*: not just detecting that something went wrong, but explaining *why* it went wrong and what to change to prevent it from happening again. My analysis thrust closes this loop by developing techniques for *specification repair*, *robustness analysis*, and *counterexample classification* for learning-enabled controllers. First, I develop constrained specification learning and repair algorithms that recover temporal operators, ordering, and time bounds from positive/negative traces while enforcing syntactical constraints (e.g., bounded formula templates, restricted operator sets) to keep the learned specifications interpretable [24]. This makes specifications dynamic artifacts evolving with data and models rather than assumptions frozen at the start of the pipeline. Second, I use falsification [5] and tolerance analysis to systematically probe reinforcement learning policies under deviations in actuation, dynamics, and sensing, producing safe operating regimes before deployment [21]. Finally, I develop counterexample-driven tools that turn raw violating trajectories into interpretable failure modes by classifying counterexamples into parametric STL classes, enabling targeted interventions guided by *which part of the specification failed and how* [4]. Together, these methods constitute a scalable workflow that makes controller improvement systematic rather than ad hoc.

C. Enforcement: runtime safety layers

Even when behavior is designed with specifications in mind, real deployments introduce distribution shift, perception errors, and modelling mismatches that can break the assumptions under which synthesis succeeds. My enforcement thrust therefore builds *runtime enforcers* from specifications that monitor execution and intervene only when necessary to prevent violations. In the setting of the vision language action models [12], I adapted the constrained decoding [10] idea from large language models to robot action generation: at each autoregressive step, we treat the policy's next-action distribution like a token distribution, run a lightweight lookahead check under dynamics and reachability assumptions, and mask candidate actions whose continuations are provably headed toward an STL violation. The model then decodes from the remaining safe action set at inference-time for arbitary STL specifications. Complementarily, we put enforcement in the loop with real visual data: we wrap a vision-driven pipeline with a Control Barrier Function [6] based safety filter that takes perception-derived obstacle/state estimates and solves a constrained QP at runtime to minimally adjust the nominal command while maintaining forward invariance of the safe set. Crucially, we evaluate this system in the wild on high speed drones with real sensor streams, lighting variation, false detections, and actuation delays showing reliable visual detect and avoid capabilities on the edge with compute constraints.

Together, these efforts position enforcement as the last line of defense in a software-engineering lifecycle for autonomy.

D. Scalable differentiable robustness as an enabling tool

A central bottleneck in specification-centric robotics is robustness throughput: most existing differentiable STL toolchains (e.g., STLCG's recurrent formulation [14]) evaluate temporal operators via recurrence, which are hard to evaluate over batches and become the rate-limiting step inside learning loops. We propose a novel technique that replaces the recurrent evaluation with a masking formulation that turns temporal operators into batched, GPU-friendly operations, making robustness (and its gradients) practical at modern scale [22]. Critically, our tool enables capabilities that were previously impractical in common toolchains: (i) large-batch robustness evaluation over long horizons, (ii) end-to-end gradient-based optimization through STL objectives, and (iii) *learning timing parameters via gradients*, expanding the expressivity and learnability of temporal specifications. This tooling supports all three thrusts.

III. Research agenda

Classical specification mechanisms assume hand-designed predicates over clean state estimates, but modern autonomy increasingly runs on learned representations. My proposed work expands specifications and their usage in two ways: (i) *representation-grounded specifications* that can be monitored directly from perceptual streams, and (ii) *specification-driven stress testing* that systematically exposes world model failure modes and turns them into actionable training and evaluation artifacts.

First, I will develop novel logical languages to make specifications perceptual rather than state-dependent. This logic will treat learned embeddings as first-class objects in the logic, defining atomic propositions over distances and relations in representation space. The goal is a monitor that consumes images and returns a robustness-like margin with calibrated uncertainty, enabling intervention under distribution shift. Crucially, this logic can be used to synthesize, analyze, and enforce task specifications with open ended alphabets.

Second, I will use these logical specifications as a test oracle for world models [9]. Because they define predicates over learned perceptual representations, they capture context-dependent safety and can be evaluated both on real sensor streams and on world-model rollouts in embedding space. The core idea is guided falsification: search for prompts/actions where the world model predicts satisfaction but execution yields a violation, pinpointing where the model fails to preserve the semantic and temporal structure that matters for safe autonomy. The resulting counterexamples become both targeted training data and a reusable evaluation suite for measuring whether world models reliably simulate safety-critical failure modes.

References

[1] Title and authors omitted for anonymous review. *In IEEE International Conference on Robotics and Automation (ICRA) Intelligent Aerial Robotics Workshop*, 2022.

[2] Title and authors omitted for anonymous review. *Robotics: Science and Systems (RSS) Safe Autonomy workshop*, 2024.

[3] Title and authors omitted for anonymous review. *Under Submission*, 2025.

[4] Title and authors omitted for anonymous review. *Under Submission*, 2026.

[5] Houssam Abbas, Georgios Fainekos, Sriram Sankaranarayanan, Franjo Ivančić, and Aarti Gupta. Probabilistic temporal logic falsification of cyber-physical systems. *ACM Trans. Embed. Comput. Syst.*, 12(2s), May 2013. ISSN 1539-9087. doi: 10.1145/2465787.2465797. URL https://doi.org/10.1145/2465787.2465797.

[6] Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications, 2019. URL https://arxiv.org/abs/1903.11199.

[7] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[8] Marie Farrell, Matt Luckcuck, and Michael 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.

[9] David Ha and Jürgen Schmidhuber. World models. 2018. doi: 10.5281/ZENODO.1207631. URL https://zenodo.org/record/1207631.

[10] Chris Hokamp and Qun Liu. Lexically constrained decoding for sequence generation using grid beam search. In Regina Barzilay and Min-Yen 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/.

[11] Yafei Hu, Quanting Xie, Vidhi Jain, Jonathan Francis, Jay Patrikar, Nikhil Varma Keetha, Seungchan Kim, Yaqi Xie, Tianyi Zhang, Shibo Zhao, et al. Toward general-purpose robots via foundation models: A survey and meta-analysis. *CoRR*, 2023.

[12] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model, 2024. URL https://arxiv.org/abs/2406.09246.

[13] Vince Kurtz and Hai Lin. Mixed-integer programming for signal temporal logic with fewer binary variables. *arXiv preprint arXiv:2204.06367*, 2022.

[14] Karen Leung, Nikos Aréchiga, and Marco Pavone. Backpropagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods. *The International Journal of Robotics Research*, page 02783649221082115, 2023.

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

[16] Anastasios Manganaris, Vittorio Giammarino, Ahmed H Qureshi, and Suresh Jagannathan. Formal methods in robot policy learning and verification: A survey on current techniques and future directions. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=DZkikdg5sl.

[17] Amir 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.

[18] Stefan Schaal. Learning from demonstration. In M.C. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1996. URL https://proceedings.neurips.cc/paper_files/paper/1996/file/68d13cf26c4b4f4f932e3eff990093ba-Paper.pdf.

[19] Title and authors omitted for anonymous review. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[20] Title and authors omitted for anonymous review. In *NASA Formal Methods Symposium*. Springer, 2024.

[21] Title and authors omitted for anonymous review. In *International Symposium on Formal Methods*. Springer, 2024.

[22] Title and authors omitted for anonymous review. *IEEE Robotics and Automation Letters*, September 2025.

[23] Title and authors omitted for anonymous review. *Robotics: Science and Systems (RSS)*, 2025.

[24] Title and authors omitted for anonymous review. In *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering*, 2025.