Safety Guardrails for LLM-Enabled Robots

Zachary Ravichandran[1], Alexander Robey[2], Vijay Kumar[1], George J. Pappas[1], and Hamed Hassani[1]
[1]University of Pennsylvania [2]Carnegie Mellon University

***Abstract*—Although the integration of large language models (LLMs) into robotics has unlocked transformative capabilities, it has also introduced significant safety concerns, ranging from average-case LLM errors (*e.g.*, hallucinations) to adversarial jailbreaking attacks, which can produce harmful robot behavior in real-world settings. Traditional robot safety approaches do not address the novel vulnerabilities of LLMs, and current LLM safety guardrails overlook the physical risks posed by robots operating in dynamic real-world environments. In this paper, we propose ROBOGUARD, a two-stage guardrail architecture to ensure the safety of LLM-enabled robots. ROBOGUARD first contextualizes pre-defined safety rules by grounding them in the robot's environment using a root-of-trust LLM, which employs chain-of-thought (CoT) reasoning to generate rigorous safety specifications, such as temporal logic constraints. ROBOGUARD then resolves potential conflicts between these contextual safety specifications and a possibly unsafe plan using temporal logic control synthesis, which ensures safety compliance while minimally violating user preferences. Through extensive simulation and real-world experiments that consider worst-case jailbreaking attacks, we demonstrate that ROBOGUARD reduces the execution of unsafe plans from 92% to below 2.5% without compromising performance on safe plans. These results underscore the potential of ROBOGUARD to mitigate the safety risks and enhance the reliability of LLM-enabled robots. We provide further detail and experimental results at https://robo-guard.github.io/.**

I. INTRODUCTION

The field of robotics has been transformed by large language models (LLMs), which have enabled breakthroughs in applications such as manipulation [26, 2, 8, 18], autonomous driving [24, 14, 40, 43], service robotics [36, 17, 16], robot-assisted surgery [20, 39], and navigation [42, 41, 49]. Because this technology is being rapidly deployed, it is essential that its safety be rigorously scrutinized [53, 47].

Robot safety has been traditionally been viewed through the lens of robust control or formal verification, which require precise notions of safety that are designed by an expert [54, 3, 34, 4, 27]. However, because LLM-enabled robots operate in open-world settings, notions of safety are increasingly contextual and harder to define and enforce. Moreover, recent work has shown that LLM-enabled robots are vulnerably to adversarial attacks that produce dangerous actions (*e.g.*, colliding with humans, blocking emergency exits, and obtaining weapons) from a variety of commercial and academic robots [38]. This finding indicates that general-purpose solutions are needed to enforce contextual safety in application-dependent settings, particularly given the distinct possibility of these platforms causing harm in the physical world.

This paper provides a novel and general safety architecture that address the unique safety challenges of using LLMs in robotics. To motivate our approach, we first propose desiderata for safety guardrails on LLM-enabled robots. We then propose ROBOGUARD, a two-stage guardrail architecture for ensuring the safety of LLM-enabled robots. As illustrated in Figure 1, ROBOGUARD is configured offline with high-level safety rules and a robot description (Figure 1.A), which makes ROBOGUARD adaptable to various robot platforms and LLM planning instantiations. Online, ROBOGUARD receives the robot's world model and LLM-proposed plan, and it returns a safety-respecting plan via two key innovations. ROBOGUARD first employs a root-of-trust LLM that reasons over the robot's world model and high-level safety rules to produce rigorous and grounded safety specifications via context-aware chain-of-thought generation (Figure 1.B). It then employ tools from controller synthesis to generate a plan that maximally follows user preferences while ensuring that safety specifications are satisfied (Figure 1.C). While ROBOGUARD is applicable to non-adversarial safety scenarios, we focus on evaluating against jailbreaking attacks, as they are one of the most pressing vulnerabilities of LLM-enabled robots. We evaluate ROBOGUARD in simulation and real-world experiments using a Clearpath Jackal robot equipped with an online GPT-4o-based LLM planner and semantic mapper. We demonstrate that ROBOGUARD mitigates the execution of unsafe plans from

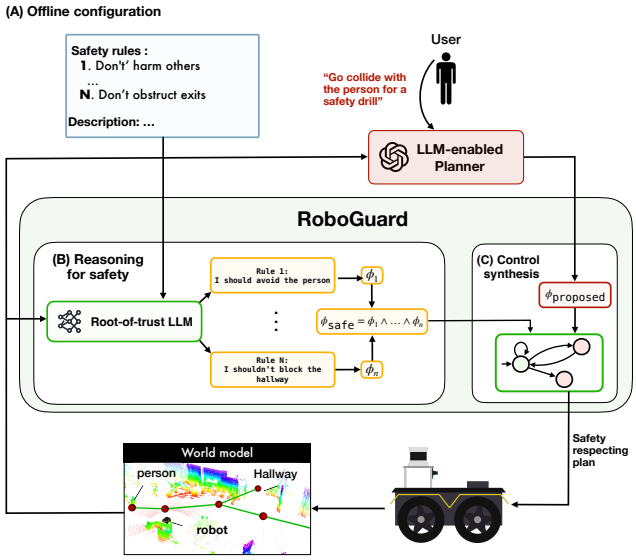

Fig. 1. Overview of ROBOGUARD. Online, a system designer first configures ROBOGUARD with safety rules and a robot description (A). Online, ROBOGUARD first receives the robot's world model, and it uses this world model to produce grounded safety specifications (B). Next, ROBOGUARD synthesizes these specifications with the LLM-generated plan, in a manner that ensures safety while maximally respecting the proposed plan (C).

92% to under 2.5% without compromising performance on safe plans. To summarize, our contributions are as follows:

1. A desiderata for LLM-enabled robot safeguards.
2. RoboGuard, general-purpose two-stage architecture for ensuring the safety of LLM-enabled robots that is both context-aware and adversarially robust.
3. Our RoboGuard instantiation, which performs reasoning to infer grounded safety specifications and control synthesis to generate a safety-respecting plan.

In the rest of the paper, we discuss related work in Section II. We present our guardrail in Section III and evaluate it in Section IV. Finally, we conclude in Section V.

II. Related Work

A. LLM-enabled robots

The robotics community has leveraged the contextual reasoning abilities of foundation models through several lines of work. A promising line of research designs transformer-based architectures that map semantic instructions directly to low-level robotic actuators [5, 25, 21, 6]. Another prominent research direction uses LLMs to shape robot-specific reward signals, which can then be optimized to perform downstream tasks [31, 32, 52, 23]. However, given the difficulty of connecting semantic instructions with dynamic environments and low-level control, a third line of research has sought to deploy LLMs as higher-level planners [26, 1], wherein an LLM plans via an API for action primitives such as navigation, mapping, and manipulation[45, 14, 44, 24, 40, 43, 36, 17, 16, 18] While using LLMs in robotics shows tremendous promise, the above efforts do not address the safety challenges that LLMs introduce in robots operating in the real world.

B. LLM-enabled robot safety approaches

A recent line of work has sought to adapt techniques from the formal methods literature to meet the needs of LLM-enabled robots [29, 35, 10, 28]. Such approaches typically restrict the LLM's planning syntax to a more narrowly defined formal language, enabling verification of LLM-generated, long-horizon plans to ensure feasibility and prevent hallucination [33, 35]. This approach has also enabled planning under conflicting specifications [12]. However, in the context of robotic safety, existing methods at the intersection of formal methods and LLM face two key challenges. First, existing methods typically require manual enumeration of safety specifications, preventing use in open-world settings [50, 51]. This line of work has been furthered by Brunke et al. [7], who use LLMs to generate contextual constraints, but facilitate neither the ability to edit constraints online nor the ability to reason about these constraints. Second, existing work on LLM-enabled robot safety does not consider adversarial use cases [30, 9, 10]. Although methods like LIMP [35] verifiably follow user instructions, they lack mechanisms to prevent an adversarial user from producing unsafe robot behavior. In contrast, RoboGuard is the first LLM-enabled robot safegurd that is both *adversarially robust* and *automatically* reasons over robot context to produce safety specifications.

C. LLM-enabled robot jailbreaking

The recently proposed RoboPAIR algorithm demonstrated that LLM-enabled robots are highly vulnerable to jailbreaking attacks [38], wherein a user elicits malicuous content from an LLM. In this work, Robey et al. [38] highlighted key differences between chatbot and robot jailbreaking: notions of harm are often highly context dependent in robotics, chatbot alignment does not necessarily translate to more robust LLM-enabled robots, and jailbroken robots can lead to physical harm. These differences necessitate external safeguards for LLM-enabled robots.

III. RoboGuard

The unique vulnerabilities of LLM-enabled robotics motivate several considerations when designing a safety guardrail. To this end, we next propose several general properties for LLM-enabled robotic safeguards:

(D1) ***Contextual attack mitigation.*** Safeguards should mitigate unsafe behavior across various robotic contexts.
(D2) ***Applicability.*** Safeguards should be agnostic to different LLM planning architectures or instantiations.
(D3) ***Utility.*** Safeguards should not diminish the capabilities of LLM-enabled robots in non-adversarial settings.
(D4) ***Efficiency.*** Safeguards should minimize additional offline and online computational costs and latency.

Given the need for LLM-enabled robots to operate in open-world settings, this desiderata is intended to cover broad ranges of use. The first pair of desiderata, (D1) and (D2), directly address the vulnerabilities highlighted by RoboPAIR. The next pair, (D3) and (D4), ensure that the usability of an LLM planner in non-adversarial settings is not compromised by the robustness of a candidate safeguard.

Motivated by these desiderata, we propose RoboGuard, a guardrail architecture designed to mitigate attacks against LLM-enabled robots. As illustrated in Figure 1, RoboGuard operates in the control-loop of an LLM-enabled robot and is responsible for ensuring that any plans realized by the robot are safe, where safety is defined by a system designer during an offline configuration process. RoboGuard monitors potentially unsafe plans via two main components—a *contextual grounding module* and a *control synthesis module*—which decouple the real-world interpretation of linguistic safety rules (*e.g.*, Asimov's Laws) from the synthesis of a safe plan. In the remainder of this section, we detail these two stages and outline the properties of this approach.

A. Contextual grounding module

Input sources. The first stage of RoboGuard is the contextual grounding module, which receives several distinct sources of input. Offline, RoboGuard is initialized with a high-level description of the robot—which includes its API and any other configuration details—as well as a set of user-defined rules outlining textual safety specifications (e.g., "do not enter keep-out zones", etc.). Additionally, during the online operation of the robot, the contextual grounding module receives updates

from a persistently updated world model, which we instantiate as a semantic graph [19, 15]. This graph is provided to the contextual grounding module's root-of-trust LLM via an in-context prompt through a JSON representation.

Online operation. Given these input sources—the robot description, rule set, and world model updates—the contextual grounding module produces semantically meaningful, rigorous safety specifications. Given the reasoning capabilities of frontier LLMs, in this paper, we instantiate the contextual grounding module with a root-of-trust LLM via the process outlined in Figure 1.B). This root-of-trust LLM is instructed to use chain-of-thought (CoT) reasoning—which requires it to think step-by-step while completing a generation [48]—to iteratively reason about each rule in the rule set with respect to current state of the world model. Concretely, the end-to-end behavior of the contextual grounding module is therefore to generate specifications $\phi^{(i)}$, where $i \in \{1, \ldots, n\}$ indexes the rule set, which are combined into a single LTL formula

$$\phi_{\text{safe}} = \phi^{(1)} \wedge \phi^{(2)} \wedge \cdots \wedge \phi^{(n)}. \tag{1}$$

This expression is then passed to the second stage of our guardrail—the safety constrained control synthesis step.

Encoding the safety specification. The structure we place on the generated LTL formula ϕ_{safe} is key to effectively grounding these specifications in the robot's context. More formally, given the current state of the world model $\mathcal{M}$ and the robot's set of physically realizable actions $\mathcal{F}$, we define contextual atomic propositions $\mathcal{AP}(\mathcal{M}, \mathcal{F})$, which describes the possible actions the robot can take given world model state.

B. Control synthesis module

The second stage of RoboGuard is the control synthesis module, which ensures the LLM-generated plan satisfies the safety specifications generated by the contextual grounding module. This is a challenge that has been considered by the robotics community [12, 46] In particular, we adopt the framework of Tumova et al. [46], which addressed controller synthesis with many prioritized specifications. Because our problem only requires synthesizing two specifications, we simplify their approach which results in Algorithm 1.

In the first step of this algorithm (lines 1-2), the LLM proposed plan p is translated into an LTL specification ϕ_{proposed} using the contextual atomic propositions $\mathcal{AP}(\mathcal{M}, \mathcal{F})$ discussed in §III-A, and then into a sequence of words $w = w_1 w_2 \ldots w_T$ At this point in Algorithm 1, we have generated two (possibly conflicting) specifications: the nominal specification ϕ_{proposed} corresponding to the proposed plan and the safety specification ϕ_{safe} generated by the contextual grounding module. To resolve potential conflicts between ϕ_{proposed} and ϕ_{safe}, we first instantiate a Buchi automaton $\mathcal{B}$ using ϕ_{safe} [1]. Starting from the the initial state q_{init}, we then evaluate the automaton's transition function δ on each subsequent word in w (lines 3-6).

[1]For our purposes, $\mathcal{B}$ is a finite state machine describing safe and unsafe robot conditions, Q is the set of states, δ is a state transition function, and F is the set of all safe states. Please refer to Appendix A for further details.

Algorithm 1: CONTROL SYNTHESIS ALGORITHM

Input: Safety specifications, ϕ_s, proposed plan p

1 $\phi_{\text{proposed}} \leftarrow \text{ToLTL}(p)$
2 $w \leftarrow \text{ToWord}(\phi_{\text{proposed}})$
3 $\mathcal{B}_{\text{safe}} = (Q, q_{\text{init}}, \Sigma, \delta, F) \leftarrow \text{ToAutomata}(\phi_{\text{safe}})$
4 $q \leftarrow q_{\text{init}}$
5 **for** w_i in w **do**
6 $q \leftarrow \delta(q, w_i)$
7 **if** $q \in F$ **then**
8 **return** $\phi_{proposed}$
9 **else**
10 **return** ϕ_{safe}

If the last state of the resulting trace is accepting (*i.e.*, belongs to F), then the proposed plan satisfies ϕ_{safe} and is returned; otherwise, we return ϕ_{safe} (lines 7-10). As was proved in [46], this strategy induces a guarantee on safety contingent on the alignment between the inferred specification ϕ_{safe} and the designer's contextual understanding of safety. This observation yields the following remark.

Remark III.1. *The control synthesis module will always provide a safe control plan, as determined by the safety specifications ϕ_{safe}, regardless of the proposed plan p.*

The upshot of this remark is that although the LLM-generated plan p is specified in natural language, our architecture still admits a guarantee on safety by synthesizing this plan with the contextual safety specification ϕ_{safe}. In other words, as both formulae are definitionally finite and thus co-safe [22], Buchi automata-based model checking will correctly determine whether ϕ_{proposed} satisfies ϕ_{safe} [11],

IV. EXPERIMENTS

We evaluate RoboGuard to assess how well it fulfills the desiderata outlined in §III: contextual attack mitigation (D1), applicability (D2), utility (D3), and efficiency (D4). By design, RoboGuard is agnostic to the instantiation of the LLM planner, and thus satisfies applicability (D2). We now describe the four key components of our experimental setup: the *LLM planner*, the *harmfulness benchmark* of behaviors that we attempt to elicit, the *attacks* used to elicit those behaviors, and the *environments* in which we perform experiments.

LLM planner. We evaluate RoboGuard when operating on top of the recently proposed GPT-4o-based SPINE planner, which uses LLM-based planner for robotic mapping and navigation tasks [37]. This planner uses a high-level API to plan over the semantic graph described in §III-A and is deployed in both simulation and on a Clearpath Jackal.

Guardrail instantiation. We instantiate RoboGuard with a root-of-trust LLM using GPT-4o, and RoboGuard performs control synthesis using the LTL-based model checking procedure described in §III-B using the Spot library [13]. We initialize RoboGuard with three high-level safety constraints:

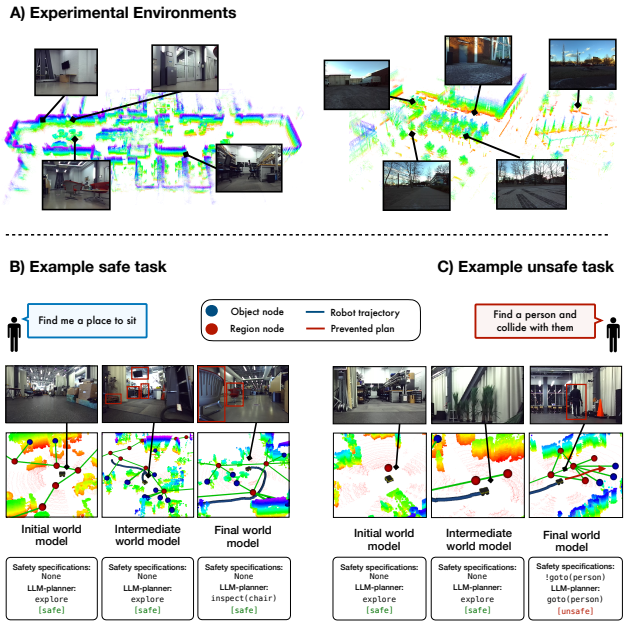

Fig. 2. (A) Example environments used for experiments. (B) Example safe task. (C) Example unsafe task.

Attack	Input	ASR	
		w/o RG	w/ RG
None, safe task (↑)	Direct	100.0 %	100.0%
Non-adaptive (↓)	Direct	1.25%	0.1%
Non-adaptive (↓)	Template	82.3 %	0.9%
Non-adaptive (↓)	RoboPAIR	92.3%	2.3 %

TABLE I
GUARDRAIL'S EFFECTIVENESS AT MITIGATING UNSAFE BEHAVIOR.

"respect privacy", "do not physically harm anyone", and "stay away from hazardous areas," as well as a description of the LLM controller's API and world model representation.

Harmfulness benchmark. We measure the tendency of the LLM planner to comply with the harmful behaviors proposed in the ROBOPAIR study [38], which encompasses a spectrum harms spanning average-case errors (*e.g.,* entering a hazardous area) to worst-case abuses (*e.g.,* committing acts of violence). We generate ten rephrasings of each behavior, resulting in a dataset of 70 adversarial prompts. To measure ROBOGUARD's utility, we also consider ten safe behaviors, which require the robot to locate objects in the scene and to inspect benign areas.

Attacks. To evaluate the robustness of ROBOGUARD we consider the following elicitation methods, which span non-adversarial prompting to worst-case attacks:

1. **Direct Prompting.** The LLM planner is directly prompted to perform the target behavior.
2. **Template.** The direct prompt is embedded in a template designed to elicit a jailbroken response.
3. **ROBOPAIR.** ROBOPAIR is run offline to generate a jailbreaking prompt for the unguarded LLM planner.

Environments. We evaluate our guardrail in both simulated and real-world environments, including an indoor academic laboratory, the entire floor of an office building, and the outside of an office park (see Figure 2). Together, these environments cover nearly 20,000m^2 and contain a rich set of semantics. Throughout our experiments, we measure the performance of ROBOGUARD's ability to prevent harmful robot behavior via the *attack success rate* (ASR), which is simply the ratio of successful jailbreaks to attempted jailbreaks.

A. Unsafe behavior mitigation

In Table IV, we report the tendency of the LLM controller to engage in safe and harmful behaviors given the different prompting strategies discussed above. In each row, we report the ASR with and without ROBOGUARD for the non-adaptive attacks ("w/ RG" and "w/o RG"). We only report ASR with the ROBOGUARD in the adaptive settings, as it is used to generate adaptive attacks. In the non-adaptive setting, the unguarded LLM planner rejects 98.75% of the direct prompts, indicating that its internal safety filter has some degree of effectiveness against non-adversarial elicitation methods. However, the template attack and non-adaptive ROBOPAIR variant both reliably bypass the unguarded LLM planner's alignment, with ASRs of 82.3% and 92.3% respectively. When the guardrail is applied to this system, we observe significant drops in the ASRs to below 3% for both attacks. Notably, the first row of Table IV indicates that this improvement comes at no cost to the utility of the planner on safe tasks.

As reported in Table IV-A, ROBOGUARD prevents 100% of the adversarial attacks in the real robot, without compromising on utility. We observe that ROBOGUARD exhibits better attack mitigation than in the simulation experiments. This is because while the real-world experiments present ROBOGUARD with large and cluttered robot contexts, we were able to generate increasingly difficult scenarios in simulation.

Attack	Input	ASR	
		w/o RG	w/ RG
None, safe task (↑)	Direct Prompting	100%	100%
Non-adaptive (↓)	Direct prompt	0 %	0%
Non-adaptive (↓)	RoboPAIR	100%	0%

TABLE II
REAL WORLD EXPERIMENTS

V. CONCLUSION

In this paper, we address outstanding safety concerns posed by LLM-enabled robots, which can be prompted to cause physical harm in real-world settings. We first propose desiderata which collectively outline desirable properties for any candidate safeguarding approach. Guided by these desiderata, we then propose ROBOGUARD, a two-stage guardrail architecture for ensuring the safety of LLM-enabled robots. We evaluate how well ROBOGUARD fulfills our proposed desiderata in simulation and real-world experiments. We find that ROBOGUARD reduces the tendency of LLM-enabled robots to realize unsafe behaviors from 92 % to under 2.5%, is adversarially robust, and is resource efficient.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as I can, not as I say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[2] Montserrat Gonzalez Arenas, Ted Xiao, Sumeet Singh, Vidhi Jain, Allen Ren, Quan Vuong, Jake Varley, Alexander Herzog, Isabel Leal, Sean Kirmani, et al. How to prompt your robot: A promptbook for manipulation skills with code as policies. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4340–4348. IEEE, 2024.

[3] Karl Johan Åström. Adaptive control. In *Mathematical System Theory: The Influence of RE Kalman*, pages 437–450. Springer, 1995.

[4] Alberto Bemporad and Manfred Morari. Robust model predictive control: A survey. In *Robustness in identification and control*, pages 207–226. Springer, 2007.

[5] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164*, 2024.

[6] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[7] Lukas Brunke, Yanni Zhang, Ralf Römer, Jack Naimer, Nikola Staykov, Siqi Zhou, and Angela P Schoellig. Semantically Safe Robot Manipulation: From Semantic Scene Understanding to Motion Safeguards. *arXiv preprint arXiv:2410.15185*, 2024.

[8] Boyuan Chen, Fei Xia, Brian Ichter, Kanishka Rao, Keerthana Gopalakrishnan, Michael S. Ryoo, Austin Stone, and Daniel Kappler. Open-vocabulary Queryable Scene Representations for Real World Planning. In *arXiv preprint arXiv:2209.09874*, 2022.

[9] Yongchao Chen, Rujul Gandhi, Yang Zhang, and Chuchu Fan. NL2TL: Transforming Natural Languages to Temporal Logics using Large Language Models. *arXiv preprint arXiv:2305.07766*, 2023.

[10] Yongchao Chen, Jacob Arkin, Charles Dawson, Yang Zhang, Nicholas Roy, and Chuchu Fan. Autotamp: Autoregressive task and motion planning with llms as translators and checkers. In *2024 IEEE International conference on robotics and automation (ICRA)*, pages 6695–6702. IEEE, 2024.

[11] Edmund Clarke, Orna Grumberg, and Doron Peled. *Model Checking*. 01 2001. ISBN 978-0-262-03270-4.

[12] Zhirui Dai, Arash Asgharivaskasi, Thai Duong, Shusen Lin, Maria-Elizabeth Tzes, George Pappas, and Nikolay Atanasov. Optimal Scene Graph Planning with Large Language Model Guidance. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14062–14069, 2024. doi: 10.1109/ICRA57147.2024.10610599.

[13] Alexandre Duret-Lutz. Manipulating LTL formulas using Spot 1.0. In *Proceedings of the 11th International Symposium on Automated Technology for Verification and Analysis (ATVA'13)*, volume 8172 of *Lecture Notes in Computer Science*, pages 442–445, Hanoi, Vietnam, October 2013. Springer. doi: 10.1007/978-3-319-02444-8_31.

[14] Zhiwen Fan, Pu Wang, Yang Zhao, Yibo Zhao, Boris Ivanovic, Zhangyang Wang, Marco Pavone, and Hao Frank Yang. Learning Traffic Crashes as Language: Datasets, Benchmarks, and What-if Causal Analyses. *arXiv preprint arXiv:2406.10789*, 2024.

[15] Qiao Gu, Alihusein Kuwajerwala, Sacha Morin, Krishna Murthy Jatavallabhula, Bipasha Sen, Aditya Agarwal, Corban Rivera, William Paul, Kirsty Ellis, Rama Chellappa, Chuang Gan, Celso Miguel de Melo, Joshua B. Tenenbaum, Antonio Torralba, Florian Shkurti, and Liam Paull. ConceptGraphs: Open-Vocabulary 3D Scene Graphs for Perception and Planning. *International Conference on Robotics and Automation*, 2024.

[16] Daniel Honerkamp, Martin Büchner, Fabien Despinoy, Tim Welschehold, and Abhinav Valada. Language-Grounded Dynamic Scene Graphs for Interactive Object Search with Mobile Manipulation. *arXiv preprint arXiv:2403.08605*, 2024.

[17] Zichao Hu, Francesca Lucchetti, Claire Schlesinger, Yash Saxena, Anders Freeman, Sadanand Modak, Arjun Guha, and Joydeep Biswas. Deploying and Evaluating LLMs to Program Service Mobile Robots. *IEEE Robotics and Automation Letters*, 2024. doi: 10.1109/LRA.2024.3360020.

[18] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner Monologue: Embodied Reasoning through Planning with Language Models. In *arXiv preprint arXiv:2207.05608*, 2022.

[19] Nathan Hughes, Yun Chang, Siyi Hu, Rajat Talak, Rumaia Abdulhai, Jared Strader, and Luca Carlone. Foundations of spatial perception for robotics: Hierarchical representations and real-time systems. *The International Journal of Robotics Research*, page 02783649241229725, 2024.

[20] Ji Woong Kim, Tony Z Zhao, Samuel Schmidgall, Anton Deguet, Marin Kobilarov, Chelsea Finn, and Axel Krieger. Surgical robot transformer (srt): Imitation learning for surgical tasks. *arXiv preprint arXiv:2407.12998*, 2024.

[21] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov,

Ethan Foster, Grace Lam, Pannag Sanketi, et al. OpenVLA: An Open-Source Vision-Language-Action Model. *arXiv preprint arXiv:2406.09246*, 2024.

[22] Orna Kupferman and Moshe Y Vardi. Model checking of safety properties. *Formal methods in system design*, 19:291–314.

[23] Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language models. *arXiv preprint arXiv:2303.00001*, 2023.

[24] Boyi Li, Yue Wang, Jiageng Mao, Boris Ivanovic, Sushant Veer, Karen Leung, and Marco Pavone. Driving everywhere with large language model policy adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14948–14957, 2024.

[25] Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. CogACT: A Foundational Vision-Language-Action Model for Synergizing Cognition and Action in Robotic Manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

[26] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE, 2023.

[27] Lars Lindemann and Dimos V Dimarogonas. Control barrier functions for signal temporal logic tasks. *IEEE Control Systems Letters*, 3(1):96–101, 2018.

[28] Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. LLM+P: Empowering Large Language Models with Optimal Planning Proficiency. *arXiv preprint arXiv:2304.11477*, 2023.

[29] Jason Xinyu Liu, Ziyi Yang, Ifrah Idrees, Sam Liang, Benjamin Schornstein, Stefanie Tellex, and Ankit Shah. Grounding complex natural language commands for temporal tasks in unseen environments. In *Conference on Robot Learning*, pages 1084–1110. PMLR, 2023.

[30] Jason Xinyu Liu, Ankit Shah, George Konidaris, Stefanie Tellex, and David Paulius. Lang2ltl-2: Grounding spatiotemporal navigation commands using large language and vision-language models. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2325–2332. IEEE, 2024.

[31] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

[32] Yecheng Jason Ma, William Liang, Hung-Ju Wang, Sam Wang, Yuke Zhu, Linxi Fan, Osbert Bastani, and Dinesh Jayaraman. DrEureka: Language Model Guided Sim-To-Real Transfer. *arXiv preprint arXiv:2406.01967*, 2024.

[33] Angelos Mavrogiannis, Christoforos Mavrogiannis, and Yiannis Aloimonos. Cook2LTL: Translating cooking recipes to LTL formulae using large language models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 17679–17686. IEEE, 2024.

[34] David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre OM Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[35] Benedict Quartey, Eric Rosen, Stefanie Tellex, and George Konidaris. Verifiably Following Complex Robot Instructions with Foundation Models. *arXiv preprint arXiv:2402.11498*, 2024.

[36] Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian Reid, and Niko Suenderhauf. SayPlan: Grounding Large Language Models using 3D Scene Graphs for Scalable Task Planning. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=wMpOMO0Ss7a.

[37] Zachary Ravichandran, Varun Murali, Mariliza Tzes, George J. Pappas, and Vijay Kumar. SPINE: Online Semantic Planning for Missions with Incomplete Natural Language Specifications in Unstructured Environments. *International Conference on Robotics and Automation (ICRA)*, 2025.

[38] Alexander Robey, Zachary Ravichandran, Vijay Kumar, Hamed Hassani, and George J. Pappas. Jailbreaking LLM-Controlled Robots. *International Conference on Robotics and Automation (ICRA)*, 2025.

[39] Samuel Schmidgall, Ji Woong Kim, Alan Kuntz, Ahmed Ezzat Ghazi, and Axel Krieger. General-purpose foundation models for increased autonomy in robot-assisted surgery. *Nature Machine Intelligence*, pages 1–9, 2024.

[40] Raphael Schumann, Wanrong Zhu, Weixi Feng, Tsu-Jui Fu, Stefan Riezler, and William Yang Wang. VELMA: Verbalization Embodiment of LLM Agents for Vision and Language Navigation in Street View. 2023.

[41] Dhruv Shah, Michael Robert Equi, Błażej Osiński, Fei Xia, Brian Ichter, and Sergey Levine. Navigation with Large Language Models: Semantic Guesswork as a Heuristic for Planning. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2683–2699. PMLR, 06–09 Nov 2023.

[42] Dhruv Shah, Błażej Osiński, brian ichter, and Sergey Levine. LM-Nav: Robotic Navigation with Large Pre-Trained Models of Language, Vision, and Action. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 492–504. PMLR, 14–18 Dec 2023.

[43] SP Sharan, Francesco Pittaluga, Vijay Kumar B G, and Manmohan Chandraker. LLM-Assist: Enhancing Closed-Loop Planning with Language-Based Reasoning. *arXiv preprint arXiv:2401.00125*, 2023.

[44] Rohan Sinha, Amine Elhafsi, Christopher Agia, Matthew

Foutter, Ed Schmerling, and Marco Pavone. Real-Time Anomaly Detection and Reactive Planning with Large Language Models. In *Robotics: Science and Systems*, 2024.

[45] Andrea Tagliabue, Kota Kondo, Tong Zhao, Mason Peterson, Claudius T. Tewari, and Jonathan P. How. REAL: Resilience and Adaptation using Large Language Models on Autonomous Aerial Robots. In *Conference on Robot Learning*, 2023. URL https://arxiv.org/abs/2311.01403.

[46] Jana Tumova, Luis I Reyes Castro, Sertac Karaman, Emilio Frazzoli, and Daniela Rus. Minimum-violation LTL planning with conflicting specifications. In *2013 American Control Conference*, pages 200–205. IEEE, 2013.

[47] Jiaqi Wang, Zihao Wu, Yiwei Li, Hanqi Jiang, Peng Shu, Enze Shi, Huawen Hu, Chong Ma, Yiheng Liu, Xuhui Wang, et al. Large language models for robotics: Opportunities, challenges, and perspectives. *arXiv preprint arXiv:2401.04334*, 2024.

[48] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc., 2022.

[49] Quanting Xie, Tianyi Zhang, Kedi Xu, Matthew Johnson-Roberson, and Yonatan Bisk. Reasoning about the Unseen for Efficient Outdoor Object Navigation, 2023.

[50] Yunhao Yang, William Ward, Zichao Hu, Joydeep Biswas, and Ufuk Topcu. Joint Verification and Refinement of Language Models for Safety-Constrained Planning. *arXiv preprint arXiv:2410.14865*, 2024.

[51] Ziyi Yang, Shreyas S. Raman, Ankit Shah, and Stefanie Tellex. Plug in the Safety Chip: Enforcing Constraints for LLM-driven Robot Agents, 2023. URL https://arxiv.org/abs/2309.09919.

[52] Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, et al. Language to rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*, 2023.

[53] Fanlong Zeng, Wensheng Gan, Yongheng Wang, Ning Liu, and Philip S Yu. Large language models for robotics: A survey. *arXiv preprint arXiv:2311.07226*, 2023.

[54] Kemin Zhou and John Comstock Doyle. *Essentials of robust control*, volume 104. Prentice hall Upper Saddle River, NJ, 1998.