Seeing, Saying, Solving: An LLM-to-TL Framework for Cooperative Robots

Dan BW Choe[1], Sundhar Vinodh Sangeetha[2], Steven Emanuel[3], Chih-Yuan Chiu[1], Samuel Coogan[1] and Shreyas Kousik[3]

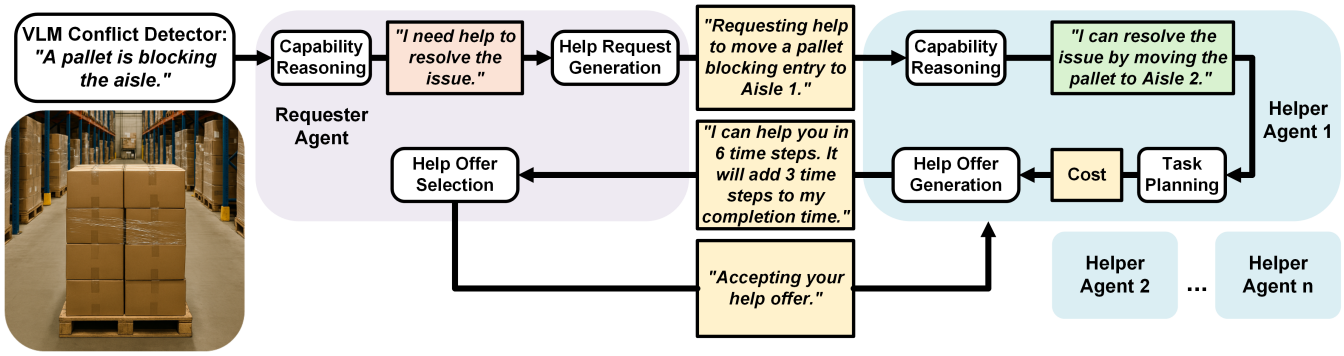

Fig. 1: Overview of the proposed framework. The requester agent encounters a conflict and broadcasts a help request to other agents if needed. A potential helper agent receives the message and reasons whether it has the required capabilities to assist. If able to assist, the potential helper agent computes the cost of helping and includes it in a help offer sent to the requester. The requester selects the help offer with the lowest cost and sends a confirmation message to the selected helper.

***Abstract*— Increased robot deployment, such as in warehousing, has revealed a need for seamless collaboration among heterogeneous robot teams to resolve unforeseen conflicts. To address this challenge, we propose a novel, decentralized framework for robots to request and provide help. The framework begins with robots detecting conflicts using a Vision Language Model (VLM), then reasoning over whether help is needed. If so, it crafts and broadcasts a natural language (NL) help request using a Large Language Model (LLM). Potential helper robots reason over the request and offer help (if able), plus information about impact to their current tasks. Helper reasoning is implemented via an LLM grounded in Signal Temporal Logic (STL) using a Backus–Naur Form (BNF) grammar to guarantee syntactically valid NL to STL translations, which are then solved as a Mixed Integer Linear Program (MILP). Finally, the requester robot chooses a helper by reasoning over impact on the overall system. We evaluate our system via experiments considering different strategies for how to choose a helper, and find that a requester robot can minimize overall time impact on the system by considering multiple help offers versus simple heuristics (e.g. selecting the nearest robot to help).**

I. Introduction

Robots ranging from drones to autonomous forklifts are increasingly common in modern warehouses, where semi-structured environments enable large-scale deployment of heterogeneous robot teams. However, the dynamic and shared nature of these spaces often leads to physical and semantic conflicts that are difficult to resolve without human intervention.

Large vision-language models (VLMs) offer a potential solution via their strengths in scene understanding and anomaly detection [1] that can enable conflict resolution in multi-robot settings [2]. However, the plans generated by VLMs cannot guarantee safety, a primary concern in shared human-robot environments. In addition, warehouse robots frequently operate under strict time constraints, such as in order fulfillment scenarios where shipments must be staged at outbound docks by designated pickup times. Without guarantees on the safety and feasibility of proposed conflict recovery plans, VLMs alone are insufficient for ensuring reliable multi-agent conflict resolution in these time-sensitive and safety-critical environments.

One path forward to solving this challenge is to apply formal methods, such as the verification of Signal Temporal Logic (STL) and Linear Temporal Logic (LTL) specifications. These can guarantee that a robot's plan satisfies constraints and have seen success in multi-robot coordination and reconfiguration tasks [3]. Recent work has also explored their integration with large language models (LLMs) to enforce safety constraints during task planning [4]. However, it remains open how to integrate these formal methods approaches with foundation model reasoning in a way that enables seamless, safe multi-agent coordination. Note, Appendix IV provides a more detailed overview of related work.

Related Work: This work lies at the intersection of foundation models and formal methods for multi-agent robot task planning. Closest to our work, [2] uses foundation models to detect, describe, and reason over conflicts and robot capabilities, but without formal guarantees. Similarly, in formal methods, [5], uses a VLM to ground spatio-temporal navigation commands in unseen environments for a single agent. In contrast to the previous works, we propose a novel

All authors are with the Georgia Institute of Technology, Atlanta, GA, USA. [1] School of Electrical and Computer Engineering. [2] School of Aerospace Engineering. [3] School of Mechanical Engineering. Corresponding author: bchoe7@gatech.edu

multi-agent framework that integrates STL to augment an LLM agent with spatial and temporal reasoning capabilities.

Contributions: As a step towards addressing the above research question, we present a framework that enables heterogeneous warehouse robots, each subject to local task specifications, to request and offer assistance in natural language, while ensuring safety and feasibility of the resulting recovery plan through formal verification. In particular, we make the following contributions:

1) We propose a method to transform natural language specifications into temporal logic specifications with a guarantee on validity (Secs. III-A and III-B).
2) We augment LLM agents with spatial and temporal reasoning capabilities, therefore making progress towards realizing guarantees from formal methods (Sec. III-C).
3) We present an experimental implementation of the proposed framework in simulation (Sec. IV).

II. Preliminaries and Problem Statement

Consider robots operating in parallel on a variety of tasks; in this work we use warehousing as our canonical example. **If a robot detects a conflict that it cannot resolve on its own, how can it request and receive help from a fellow robot while ensuring safety and minimizing impact on the overall system?** We now formalize these notions.

Notation: The reals are $\mathbb{R}$, natural numbers are $\mathbb{N}$, and integers are $\mathbb{Z}$. To allow mathematically describing functions operating on natural language, we denote $\mathbb{L}$ as a set of all natural language utterances. Subscripts are indices and superscripts are labels. We use $\leftarrow$ to denote the output of NL reasoning.

World and Robots: We consider n robots operating in a discrete time grid world, so robot i is at $x_i(t) \in W \subset \mathbb{Z}^2$ at time $t = 0, 1, \cdots, H$, where $H \in \mathbb{N}$ is a finite time horizon. Each cell is either free space or a static obstacle, a common setup in modeling MRS [6]; we denote all free cells as $F \subseteq W$. Multiple robots can occupy a free cell concurrently ($x_i(t) = x_j(t) \in F$ is allowed); we assume local coordination and collision avoidance are handled by lower-level planning and control. Finally, each robot has an NL representation of its own capabilities $c_i \in \mathbb{L}$ (e.g., "can lift pallets" or "max speed 1 m/s")

Tasks: Each robot has assigned tasks represented as an STL specification φ_i and path X_i for robot i. All robots must also obey a global specification[1] φ^{g}. We assume each robot's tasks are feasible and expressed in valid STL. For any STL expression φ, we define $T(\varphi) \in \mathbb{N}$ as the makespan, or minimum duration required to satisfy φ.

Communications: Robots can send and receive natural language messages $m \in \mathbb{L}$ instantaneously and error-free. Denote $m_{i \to j}$ as a message from robot i to robot j, and a broadcast message as $m_{i \to \mathcal{R}}$, where $\mathcal{R} = \{0, 1, \cdots, n\}$ denotes all robots.

Code available at: `https://github.com/dchoe1122/seeing_saying_solving`

[1]An example can be found in Appendix I-A (e.g., actuation limits and obstacles)

Conflicts: Each robot can detect a conflict (e.g., using VLM conflict detector as in Fig. 1), which we denote as a tuple $(m^{\mathrm{c}}, x^{\mathrm{c}}, \tau^{\mathrm{c}}) \in \mathbb{L} \times W \times \mathbb{N}$ containing a natural language description, a location, and a duration required to resolve it. We assume x^{c} is resolved when a helper robot enters the same cell as the requester for at least τ^{c} time steps.

Problem. *Suppose robot i detects a conflict $(m^{\mathrm{c}}, x^{\mathrm{c}}, \tau^{\mathrm{c}})$. Without using centralized task assignment, we seek to identify another robot $j \in \mathcal{H}_i$, where $\mathcal{H}_i = \mathcal{R} \setminus \{i\}$ is the set of candidate helper robots, and plan its motion such that it resolves the conflict while minimizing the increase in the total makespan across all robots. That is, create φ_j such that $x_j(t) = x^{\mathrm{c}}$ for all $t' \in [t, t + \tau^{\mathrm{c}}]$ while minimizing $\sum_{i \in \mathcal{R}} T(\varphi_i)$.*

III. Proposed Method

In our framework, a robot needing help (*requester*) broadcasts a help request to the multi-robot system where each robot evaluates its capabilities and availability. Then, any robot able to assist (*helper*) formulates a help offer. Prior to sending back the offer to the *requester*, each *helper* translates the help offer to STL and solves for the optimal path plan that minimizes total time impact to the system. Once solved, the *helper* responds with the help offer along with the projected time impact to the system. Finally, the *requester* chooses the *helper* with the lowest impact by a help confirmation message. Fig. 1 illustrates our framework.

To proceed, Sec. III-A covers NL processing, Sec. III-B explains our NL to STL conversion, and Sec. III-C details solving for optimal paths.

A. Generating Help Requests, Offers, and Confirmations

We implement a multi-robot communication protocol for collaborative conflict resolution using Large Language Models. This protocol involves three capabilities: send, receive and broadcast. We define three types of messages: help requests, help offers, and help confirmations[2].

Help requests are broadcast messages sent by a robot facing a conflict after determining that it requires help, by reasoning over the conflict $(m^{\mathrm{c}}, x^{\mathrm{c}}, \tau^{\mathrm{c}})$ and its own capabilities c_i, conditioned on a prompt $p^{\mathrm{c}} \in \mathbb{L}$:

$$m^{\mathrm{r}}_{i \to \mathcal{H}_i} \leftarrow \mathrm{NLR}(m^{\mathrm{c}}, x^{\mathrm{c}}, \tau^{\mathrm{c}}, c_i \mid p^{\mathrm{c}}), \tag{1}$$

where NLR abstractly represents NL reasoning, implemented with foundation models. Help requests describe the scene, location, what is required for the conflict to be resolved, and why the robot cannot resolve the conflict. To ensure conflict location and requester capabilities are included in help requests, we use constrained generation [7].

Help offers are sent by potential helpers in response to a help request. To generate help offers, each robot j reasons over its help request, its own location, and its capabilities:

$$m^{\mathrm{o}}_{j \to i} \leftarrow \mathrm{NLR}(m^{\mathrm{r}}_{i \to \mathcal{H}_i}, x_j, c_j \mid p^{\mathrm{h}}). \tag{2}$$

We again use constrained generation to ensure help offers

[2]An example help request is in Appendix II-A and help offer in Appendix II-B.

include information about capabilities.

In addition, the help offer includes the duration it will take robot j to provide help τ_j^{h} (i.e., how long does robot i need to wait?), and the additional time to help relative to completing its original tasks τ_j^{new} (i.e., how much is robot j impacted?). We compute these durations below in Sec. III-C.

Finally, the requester confirms the lowest-cost help request. That is, it finds $j^\star = \arg\min_j (\tau_j^{\text{h}} + \tau_j^{\text{new}})$, then sends $m^{\text{s}}_{i\to j^\star}$ = "accept" and $m^{\text{s}}_{i\to j}$ = "reject" for all $j \neq j^\star$.

B. Translating Help Proposals to STL Specifications

We draw from a body of recent work on the transformation of natural language tasks and specifications into temporal logic formulas using large language models [8]–[10]. We enhance these natural language to temporal logic methods by defining a Backus-Naur form (BNF) grammar, a notation system for defining formal languages that can be used to constrain the output of LLMs [11]. Specifically, defining a BNF grammar for STL involves defining unary and binary temporal and boolean operators, allowed predicates, and text representations of temporal logic relations. These enforce the syntactic validity of generated temporal logic formulas, allowing us to directly feed LLM output into an STL solver[3]. Ultimately, we generate an STL specification by reasoning over the help offer message and conflict location, conditioned on a prompt p^{STL} and subject to the BNF grammar:

$$\varphi_j^{\text{h}} \leftarrow \text{NLR}(m^{\text{o}}_{j\to i}, x^{\text{c}} \mid p^{\text{STL}}, \text{BNF}). \tag{3}$$

C. Solving for Optimal Robot Paths

Suppose each candidate helper robot $j \in \mathcal{H}_i$ has an original task specification φ_j^{orig} and has found a corresponding optimal path $X_j^{\text{orig}} = \{x_j(t) \mid t = 0, \cdots, T(\varphi_j^{\text{orig}})\}$ by minimizing Manhattan distance[4] $D(X_j^{\text{orig}})$ and makespan $T(\varphi_j^{\text{orig}})$ while always obeying global specifications:

$$X_j^{\text{orig}} = \arg\min_X \quad D(X) + T(\varphi_j^{\text{orig}}) \tag{4a}$$

$$\text{s.t.} \quad \Diamond \varphi_j^{\text{orig}} \wedge \Box \varphi^{\text{g}}. \tag{4b}$$

Once the help offer $m^{\text{o}}_{j\to i}$ is translated into φ_j^{h} as in (3), robot j computes an updated path that both helps and completes the original tasks:

$$X_j^{\text{new}} = \arg\min_X \quad D(X) + T(\varphi_j^{\text{new}}) + T(\varphi_j^{\text{h}}) \tag{5a}$$

$$\text{s.t.} \quad \varphi_j^{\text{new}} = \Diamond \varphi_j^{\text{h}} \wedge \Diamond \varphi_j^{\text{orig}} \wedge \Box \varphi^{\text{g}}. \tag{5b}$$

Importantly, we include $T(\varphi_j^{\text{h}})$ in the cost to let the helper robot minimize how long the requester must wait.

We formulate (4) and (5) as Mixed Integer Linear Programs (MILPs) by defining predicates in the STL formula as binary variables over discrete time steps, and solve them with Gurobi [12].

Finally, when crafting a help offer, robot j reports the total time it will take to help *and* the additional time to help relative to its original tasks:

$$\tau_j^{\text{h}} = T(\varphi_j^{\text{h}}) \tag{6}$$

$$\tau_j^{\text{new}} = T(\varphi_j^{\text{new}}) - T(\varphi_j^{\text{orig}}). \tag{7}$$

[3]Details on how we encode STL specifications can found in the appendix.

[4]We use Manhattan distance to capturing the number of movements in a grid world and improve computation time over solving a Mixed Integer Quadratic Program (MIQP) with Euclidean Distances.

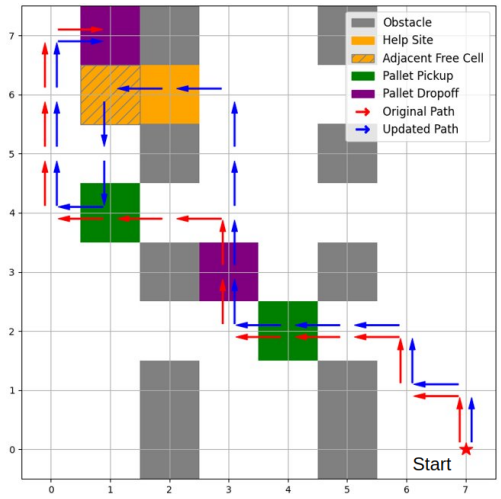

Fig. 2: An example of a reconfigured path. The help site is reached in 11 time-steps, extending the original path by 4 time-steps for a total cost $\tau_j^{\text{h}} + \tau_j^{\text{new}} = 15$ time-steps.

IV. Experiments

A. Experiment 1: Natural Language to STL

In this experiment, we evaluate our method for transforming natural language (NL) to temporal logic (TL) using BNF constrained generation. We implement our method with Gemma 3 27B LLM using the `llama.cpp` library for BNF constrained generation. The BNF grammar is included in the LLM prompt and as a constraint[5].

Hypothesis: We hypothesize that our method demonstrates comparable translation accuracy to existing baselines on a large and diverse data set of NL-TL pairs, with a strict guarantee on the validity of TL formulas, and running on a significantly smaller LLM (Gemma 3) which can be deployed onboard a robot with a single consumer GPU. This experiment is designed to test our NL to TL approach in a broad, previously unseen domain.

Experiment Design: To benchmark NL to TL translation performance, we use the dataset introduced by [13], which consists of 7,500 natural language and linear temporal logic (LTL) pairs in the context of navigation tasks.

We compare our method to several ablated variants and to a GPT-4 baseline presented in [8]. Each method is evaluated with 5 and 20 few shot examples, with 500 NL-TL pairs randomly sampled from the dataset. Each evaluation is rerun 3 times, with resampled few-shot examples and NL-TL pairs. In our approach, both the BNF grammar and few shot examples are included in the prompt, and the BNF grammar is enforced during decoding using `llama.cpp`'s constrained generation. Results with constrained generation

[5]An example BNF grammar can be found in Appendix I-C.

are not presented for GPT-4, as this capability does not exist for this LLM. The ablations include variants without the grammar constraint, without the grammar prompt, and with combinations thereof, allowing us to isolate the contributions of each component of our method.

Metrics: We measure Accuracy as percentage of generated LTL formulas logically equivalent to true LTL formulas from the dataset and Validity as percentage of generated formulas that are syntactically correct. Logical equivalence and syntactic correctness of LTL formulas are checked using the `Spot` library [14].

Results and Discussion: The results are summarized in Table I. Our method achieves 100% formula validity in all cases, confirming our hypothesized guarantee. Additionally, we show similar translation accuracy in the unseen navigation dataset with a significantly smaller LLM. Notably, including the BNF grammar in the prompt without the grammar constraint achieves higher accuracy than constrained generation, illustrating a drawback of this approach. Addressing this issue is the subject of current works, which have proposed new constrained decoding algorithms [15]. See Appendix III for full results, including inference time and additional ablations.

# Ex.	Method Variant	Accuracy (%)	Validity (%)
5	Gemma F + P + C (Ours)	56.80 ± 8.76	**100.0** ± 0.0
	F + P	**63.00** ± 5.87	99.8 ± 0.45
	F	56.80 ± 4.76	99.0 ± 1.22
	GPT-4 F + P	64.10 ± 8.79	100.0 ± 0.0
20	Gemma F + P + C (Ours)	76.73 ± 2.93	100.0 ± 0.0
	F + P	**89.73** ± 2.37	**100.0** ± 0.0
	F	86.00 ± 0.40	100.0 ± 0.0
	GPT-4 F + P	92.73 ± 3.00	100.0 ± 0.0
	F (baseline)	87	–

TABLE I: Ablation study on NL to TL with varying number of few-shot examples. F, P, and C refer to few shot prompting, inclusion of the BNF grammar in the prompt, and the BNF grammar constrained generation respectively.

B. Experiment 2: Mobile Robot Blocked by a Pallet

Our second experiment is a case study in which forklift robots respond to a help request from a mobile robot prompted by the scene description m^{c} = "A pallet is blocking the entrance to the picking aisle." Each forklift robot $j \in \mathcal{H}$ then evaluates whether it can handle this additional task (moving the obstructing pallet) on top of its existing pick-and-place (PNP)[6] jobs by updating its STL specification $\varphi_j^{\mathrm{orig}}$. The help task, φ_j^{h}, is similarly encoded as a PNP job where the forklift must travel to the help site, pick up the obstructing pallet and place the pallet in the nearest free cell $x^{\mathrm{adj}} = \arg\min_x \{D(x, x^{\mathrm{c}}) \mid x \in F\}$.

Hypothesis: We hypothesize that, by optimizing the helper's path to minimize both the requester's waiting time and the additional time incurred by the helper when providing assistance, we can significantly reduce the total extra time across all agents compared to common heuristic-based approaches.

[6]Our STL specification of a PNP job is in Appendix I-B.

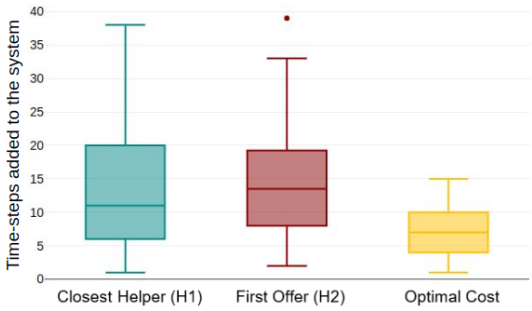

Fig. 3: Comparison of total time-steps added to the system under heuristics and the optimal solution. The closest helper (H1) coincided with the optimal solution in 42% of trials while the first offer (H2) was optimal in only 25% of trials.

Experiment Design: We conducted 100 simulation trials with randomly chosen help-site locations. In each trial, six forklift robots are spawned with random initial positions. Each forklift has two randomly assigned PNP tasks. Obstacles (including the blocking pallet) remain fixed in known locations.

We compare with two heuristic baselines: (H1) choosing the forklift robot closest to the help site, and (H2) selecting the first forklift that returns a feasible solution (i.e., the robot whose Gurobi solver finishes first).

Results and Discussion: Fig. 3 summarizes our findings. Our *optimal cost* approach on average provides 26% and 40% efficiency gains over H1 and H2 respectively. We observe that the interdependencies in PNP jobs make distance-based heuristics (H1) suboptimal, consistent with previous literature in operations research [16]. Furthermore, due to the NP-Hardness of Mixed Integer Programs [17], faster compute time does not imply global optimality [18].

V. Conclusion and Future Work

We have presented a novel framework that takes in scene descriptions from VLMs, identifies conflicts, and triggers help requests. Then, the help task is converted into a set of STL constraints, generated using a BNF grammar, which are solved as a MILP to minimize the additional time incurred by all agents.

We evaluated our approach in two scenarios. First, we benchmarked our method to translate natural language to TL on a large-scale navigation dataset, and showed comparable accuracy to existing methods with improved TL validity on small, local LLMs. Second, in a case study with forklifts and a blocked mobile robot, our method showed clear improvements compared to heuristic approaches by optimizing for total added time, resulting in more efficient system-level performance.

For future work, we intend to explore scenarios where the VLM scene descriptions are uncertain or incomplete regarding the need for assistance. Furthermore, we plan to strengthen our framework with formal safety guarantees at runtime by considering methods such as Control Barrier Functions (CBFs).

References

[1] R. Sinha, A. Elhafsi, C. Agia, M. Foutter, E. Schmerling, and M. Pavone, "Real-time anomaly detection and reactive planning with large language models," *arXiv preprint arXiv:2407.08735*, 2024.

[2] Y. Kato, T. Yoshida, Y. Sueoka, *et al.*, "Design of a multi-robot coordination system based on functional expressions using large language models," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2024, pp. 3447–3454.

[3] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local ltl specifications," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 218–235, 2015.

[4] Z. Yang, S. S. Raman, A. Shah, and S. Tellex, "Plug in the safety chip: Enforcing constraints for llm-driven robot agents," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2024, pp. 14 435–14 442.

[5] J. X. Liu, A. Shah, G. Konidaris, S. Tellex, and D. 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)*, IEEE, 2024, pp. 2325–2332.

[6] R. Stern, N. Sturtevant, A. Felner, *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 10, 2019, pp. 151–158.

[7] L. Beurer-Kellner, M. Fischer, and M. Vechev, "Prompting is programming: A query language for large language models," *Proceedings of the ACM on Programming Languages*, vol. 7, no. PLDI, pp. 1946–1969, 2023.

[8] Y. Chen, R. Gandhi, Y. Zhang, and C. Fan, "Nl2tl: Transforming natural languages to temporal logics using large language models," *arXiv preprint arXiv:2305.07766*, 2023.

[9] F. Fuggitti and T. Chakraborti, "Nl2ltl–a python package for converting natural language (nl) instructions to linear temporal logic (ltl) formulas," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 2023, pp. 16 428–16 430.

[10] A. Mavrogiannis, C. Mavrogiannis, and Y. Aloimonos, "Cook2ltl: Translating cooking recipes to ltl formulae using large language models," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2024, pp. 17 679–17 686.

[11] B. Wang, Z. Wang, X. Wang, Y. Cao, R. A Saurous, and Y. Kim, "Grammar prompting for domain-specific language generation with large language models," *Advances in Neural Information Processing Systems*, vol. 36, pp. 65 030–65 055, 2023.

[12] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, 2024.

[13] C. Wang, C. Ross, Y.-L. Kuo, B. Katz, and A. Barbu, "Learning a natural-language to ltl executable semantic parser for grounded robotics," in *Conference on Robot Learning*, PMLR, 2021, pp. 1706–1718.

[14] Alexandre Duret-Lutz, E. Renault, M. Colange, *et al.*, "From Spot 2.0 to Spot 2.10: What's new?" In *Proceedings of the 34th International Conference on Computer Aided Verification (CAV'22)*, ser. Lecture Notes in Computer Science, vol. 13372, Springer, Aug. 2022, pp. 174–187.

[15] L. Beurer-Kellner, M. Fischer, and M. Vechev, "Guiding llms the right way: Fast, non-invasive constrained generation," *arXiv preprint arXiv:2403.06988*, 2024.

[16] R. De Koster, T. Le-Duc, and K. J. Roodbergen, "Design and control of warehouse order picking: A literature review," *European journal of operational research*, vol. 182, no. 2, pp. 481–501, 2007.

[17] J. Hartmanis, "Computers and intractability: A guide to the theory of np-completeness (michael r. garey and david s. johnson)," *Siam Review*, vol. 24, no. 1, p. 90, 1982.

[18] C. Belta and S. Sadraddini, "Formal methods for control synthesis: An optimization perspective," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, no. 1, pp. 115–140, 2019.

[19] V. Kurtz and H. Lin, "Mixed-integer programming for signal temporal logic with fewer binary variables," *IEEE Control Systems Letters*, vol. 6, pp. 2635–2640, 2022.

[20] K. Garg, S. Zhang, J. Arkin, and C. Fan, "Foundation models to the rescue: Deadlock resolution in connected multi-robot systems," *arXiv preprint arXiv:2404.06413*, 2024.

[21] J. Wang, G. He, and Y. Kantaros, "Probabilistically correct language-based multi-robot planning using conformal prediction," *IEEE Robotics and Automation Letters*, 2024.

[22] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus, "Optimality and robustness in multi-robot path planning with temporal logic constraints," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 889–911, 2013.

[23] P. Yu, G. Fedeli, and D. V. Dimarogonas, "Reactive and human-in-the-loop planning and control of multi-robot systems under ltl specifications in dynamic environments," in *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*, IEEE, 2023, pp. 1862–1867.

[24] J. Ren, H. Miller, K. M. Feigh, S. Coogan, and Y. Zhao, "Ltl-d*: Incrementally optimal replanning for feasible and infeasible tasks in linear temporal logic specifications," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2024, pp. 4495–4502.

APPENDIX I
ENCODING STL SPECIFICATIONS

We propose a lightweight interface to express navigational tasks as STL specifications. Inspired by `stlpy` [19], our method emphasizes extracting *spatial* propositions and decoupling *temporal* constraints. We detail our approach in the appendix, and provide an illustrative example here.

Consider encoding the sentence "*Visit Aisle 1 at least once before time T while avoiding obstacles*,". The corresponding STL expression is:

$$\varphi^{\text{spec}} = \square_{[0,H]} \neg \text{Obstacle} \wedge \diamond_{[0,T]} \text{Aisle1} \tag{8}$$

where Aisle1 and Obstacle are atomic propositions (AP) tied to the corresponding coordinates in the environment, and become true whenever the agent is inside the region.

We then specify the time horizon T independently when setting up the optimization problem. The resulting code snippet is:

```
model.spec = F(aisle1) & G(NOT(obstacle))
model.T = T
```

A. Global STL Specification

Each robot in our framework is subject to a global specification of safety and actuation constraints. For example, all ground vehicles must avoid obstacle cells and can only move one grid space in any of the cardinal directions. Therefore all ground robots in the system are subjected to:

$$\varphi^{\text{global}} = \square(\neg \text{Obstacle} \wedge \mathcal{L}^{\text{actuation}}) \tag{9}$$

B. Pick-and-Place (PNP) Tasks as STL Specifications

We create a PNP specification as

$$\varphi_1 = (\neg x^{\text{place}} \, \mathbf{U}_{[0,H]} \, x^{\text{pick}})$$
(Robot must pickup the pallet before placing it)
$$\varphi_2 = \square_{[0,H]}(x^{\text{pick}} \rightarrow (\neg x^{\text{others}} \mathbf{U}_{[0,H]} x^{\text{place}})$$
(Pallet picked up must be placed before starting other tasks)
$$\varphi_3 = \diamond_{[0,H]} x^{\text{place}}$$
(Pallet is eventually placed within time horizon H)
$$\varphi_{\text{pnp}}(x^{\text{place}}, x^{\text{pick}}) = \varphi_1 \wedge \varphi_2 \wedge \varphi_3$$

C. Example LTL BNF Grammar

```
root ::= ltl-depth-2
ltl-depth-2 ::= proposition
             | "~(" ltl-depth-1 ")"
             | ltl-depth-1 binary-op ltl-depth-1
             | unary-op "(" ltl-depth-1 ")"

ltl-depth-1 ::= proposition
             | "~(" proposition ")"
             | proposition binary-op proposition
             | unary-op "(" proposition ")"

proposition ::= 'go_to_flag' | 'pass_by_trash_bin'

binary-op ::= "␣&␣" | "␣|␣" | "␣->␣" | "␣U␣"
# '&' (and): both propositions must be true
# '|' (or): at least one predicate must be true
# '->' (implies): if first predicate is true, then second predicate must be true
# 'U' (until): first predicate must be true at least until second predicate is true

unary-op ::= globally | eventually
globally ::= "G"
# Predicate must always be true at every timestep

eventually ::= "F"
# Predicate must be true at some time in the future
```

APPENDIX II
NATURAL LANGUAGE

A. Example Help Request

```
"A pallet is blocking the aisle at location (1, 4). Assistance is required to move the
pallet. Capabilities needed: 'lift' or 'push_heavy_objects' to clear the obstruction."
Reasoning:
1. Why I can't resolve this alone: My "move" capability only allows navigation, not
manipulating objects like pallets.
2. Required capabilities: A robot with "lift" or "push_heavy_objects" is needed to
physically relocate the pallet.
3. Request clarity: Specifying the obstacle type (pallet), location (1, 4), and
required capabilities ensures helpers can assess their ability to assist.
```

B. Example Help Offer

```
Assistance Proposal:
1. Capabilities Utilized:
- Lift Pallet: The robot can securely lift and relocate the pallet.
- Move: The robot can navigate to the target location and transport the pallet.

2. Effort Analysis:
- Task Complexity: Low to moderate. Moving a single pallet is within standard
operational parameters.
- Navigation: Direct path to (4, 0) is assumed clear (no additional obstacles
mentioned).

Location: (4, 0)
The robot will move to the specified location, lift the pallet, and relocate it to an
unoccupied storage zone to clear the aisle.
```

APPENDIX III
EXPERIMENT I FULL RESULTS

Full results of the ablation study on NL to TL with varying number of few-shot examples are presented in Appendix III. F, P, and C refer to few shot prompting, inclusion of the BNF grammar in the prompt, and the BNF grammar constrained generation respectively. For GPT-4, inference time is the time from an API call to the model to a response, which includes network latency in addition to LLM inference time.

# Examples	Method Variant	Accuracy (%)	Validity (%)	Inference Time (ms)
5	Gemma F + P + C (Ours)	56.80 ± 8.76	100.0 ± 0.0	801.6 ± 36.2
	F + P	63.00 ± 5.87	99.8 ± 0.45	583.4 ± 28.2
	F	56.80 ± 4.76	99.0 ± 1.22	410.2 ± 9.2
	F + C	49.40 ± 5.59	100.0 ± 0.0	629.4 ± 26.1
	GPT-4 prompt, no constraint	64.10 ± 8.79	100.0 ± 0.0	803.7 ± 30.3
20	Gemma F + P + C (Ours)	76.73 ± 2.93	100.0 ± 0.0	1191.7 ± 14.1
	F + P	89.73 ± 2.37	100.0 ± 0.0	974.9 ± 2.5
	F	86.00 ± 0.40	100.0 ± 0.0	414.9 ± 3.9
	F + C	72.27 ± 2.89	100.0 ± 0.0	633.5 ± 8.5
	GPT-4 F + P	92.73 ± 3.00	100.0 ± 0.0	2736.9 ± 25.2
	F (baseline)	87	–	–

APPENDIX IV
RELATED WORK

This work lies at the intersection of foundation models and formal methods for multi-agent robot task planning.

A. Foundation Models in Multi-Robot Settings

Foundation models have been specifically adapted for failure resolution in multi-robot systems (MRS). Closest to our work, [2] proposes a heterogeneous MRS where each robot uses a VLM to detect and describe impediments to task progress and an LLM to reason over robot capabilities to determine when to ask for and offer help. However, this approach relies on foundation models for reasoning, providing no safety or optimality guarantees; we adopt a similar framework but leverage formal logic to address these limitations. Conflicts between robots, as opposed to conflicts with the environment, can also be resolved using foundation models. For example, VLMs can resolve deadlock in a connected MRS by choosing a leader agent and a set of waypoints for the leader using descriptions of the task and environment [20]; note that our work focuses on environment conflicts.

Finally, foundation model-driven MRS are typically assessed on success rates of various tasks, but have not been studied extensively in terms of efficiency or optimality. An attempt in this direction uses conformal prediction on an LLM's built-in confidence score to determine whether multi-robot task and motion plans are probabilistically correct [21]. In contrast, our work uses formal logic and a convex solver to enable a foundation model to assess optimality.

B. Formal Methods in Robotics

Formal methods, such as Temporal Logic (TL), offer robust guarantees for MRS task specifications, but often assume static environments and specifications [22]. Recent literature attempts to tackle this limitation by making LTL planning more robust and dynamic. For example, one can use multi-robot and human-in-the-loop (HIL) planning where robots share new conflicts and synthesize alternate trajectories to satisfy their LTL specifications [23]. Similarly, but in a single-agent scenario, [24] takes an incremental optimal replanning strategy for both cases where the satisfaction of the LTL specification is feasible and infeasible. Finally, [5] has the closest resemblance to our work, using a VLM to ground spatio-temporal navigation commands in unseen environments for a single agent case. In contrast to the previous works, we propose a novel multi-agent framework that integrates STL to augment an LLM agent with spatial and temporal reasoning capabilities.