Grounding Complex Natural Language Commands for Temporal Tasks in Unseen Environments

Anonymous Author(s)
Affiliation
Address
email

Abstract: Grounding navigational commands to linear temporal logic (LTL) leverages its unambiguous semantics for reasoning about long-horizon tasks and verifying the satisfaction of temporal constraints. Existing approaches require training data from the specific environment and landmarks that will be used in natural language to understand commands in those environments. We propose Lang2LTL, a modular system and a software package, that leverages large language models (LLMs) to ground temporal navigational commands to LTL specifications in environments without prior language data. We comprehensively evaluate Lang2LTL for five well-defined generalization behaviors. Lang2LTL demonstrates the state-of-the-art ability of a single model to ground navigational commands to diverse temporal specifications in 21 city-scaled environments. Finally, we demonstrate a physical robot using Lang2LTL can follow 52 semantically diverse navigational commands in two indoor environments. [1]

Keywords: language grounding, temporal reasoning, robot navigation

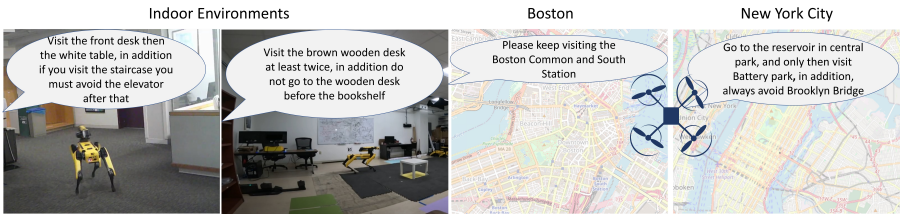


Figure 1: Lang2LTL can ground complex navigational commands in household and city-scaled environments without retraining.

1 Introduction

Natural language enables humans to express complex temporal tasks, like "Go to the grocery store on Main Street at least twice, but only after the bank, and always avoid the First Street under construction." Such commands contain goal specifications and temporal constraints. A robot executing this command must identify the bank as its first subgoal, followed by two separate visits to the grocery store, and visiting First Street is prohibited throughout the task execution. Linear temporal logic [1] provides an unambiguous target representation for grounding a wide variety of temporal commands.

Prior work of grounding natural language commands to LTL expressions for robotic tasks covered a limited set of LTL formulas with short lengths, and required retraining for every new environment [2, 3, 4, 5]. In contrast, some recent approaches proposed leveraging LLMs to directly generate robot policies [6, 7, 8, 9]. The LLM output is itself a formal language that must be interpreted by an external

[1] Code and datasets are at http://lang2ltl-corl23.conferencesubmissions.net

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

user-defined program. Such approaches cannot solve many complex temporal tasks considered in this work. Another advantage of using a formal language like LTL is that existing work on automated planning with LTL specifications provides strong soundness guarantees.

In this paper, we propose Lang2LTL, a system capable of translating language commands to grounded LTL expressions that are compatible with a range of planning and reinforcement learning tools [10, 11, 12, 13, 14]. Lang2LTL is a modular system that separately tackles referring expression recognition, grounding these expressions to real-world landmarks, and translating a lifted version of the command to obtain the grounded LTL specification. This modular design allows Lang2LTL to transfer to novel environments without retraining, provided with a semantic map.

We formally define five generalization behaviors that a learned language grounding model must exhibit, and comprehensively evaluate Lang2LTL's generalization performance on a novel dataset containing 2125 semantically unique LTL formulas corresponding to 47 LTL formula templates. Lang2LTL showed the state-of-the-art capabilities of grounding navigational commands in 21 cities using semantic information from a map database in a zero-shot fashion. Finally, we demonstrated that a physical robot equipped with Lang2LTL was able to follow 52 semantically diverse language commands in two different indoor environments, provided with semantic maps.

2 Preliminaries

Large Language Models: Autoregressive large language models (LLMs) are producing SoTA results on a variety of language-based tasks due to their general-purpose language modeling capabilities. LLMs are large-scale transformers [15] pretrained to predict the next token given a context window [16, 17]. In this work, we used the GPT series models [18, 19] and the T5-Base model [20].

Temporal Task Specification: Linear temporal logic (LTL) [1] has been the formalism of choice for expressing temporal tasks for a variety of applications including planning and reinforcement learning [10, 21, 22, 11, 23, 13], specification elicitation [3, 24, 25, 26, 2], and assessment of robot actions [27]. The grammar of LTL extends propositional logic with a set of temporal operators defined as follows:

$$\varphi := \alpha \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \mathbf{X}\varphi \mid \varphi_1 \ \mathbf{U} \ \varphi_2 \tag{1}$$

An LTL formula φ is interpreted over a discrete-time trace of Boolean propositions, $\alpha \in AP$ that maps an environment state to a Boolean value. $\varphi,\ \varphi_1,\ \varphi_2$ are any valid LTL formulas. The operators $\neg$ (not) and $\vee$ (or) are identical to propositional logic operators. The temporal operator $\mathbf{X}$ (next) defines the property that $\mathbf{X}\varphi$ holds if φ holds at the next time step. The binary operator $\mathbf{U}$ (until) specifies an ordering constraint between its two operands. The formula $\varphi_1 \ \mathbf{U} \ \varphi_2$ holds if φ_1 holds at least until φ_2 first holds, which must happen at the current or a future time. In addition, we use the following abbreviated operators, $\wedge$ (and), $\mathbf{F}$ (finally or eventually), and $\mathbf{G}$ (globally or always), that are derived from the base operators. $\mathbf{F}\varphi$ specifies that the formula φ must hold at least once in the future, while $\mathbf{G}\varphi$ specifies that φ must always hold.

We developed Lang2LTL based on a subset of specification patterns commonly occurring in robotics [28]. Appendix A lists the specification patterns and their interpretations.

Planning with Temporal Task Specification: Every LTL formula can be represented as a Büchi automaton [29, 30] thus providing sufficient memory states to track the task progress. The agent policy can be computed by any MDP planning algorithm on the product MDP of the automaton and the environment [10, 21, 22, 11, 12, 14]. Lang2LTL is compatible with any planning or reinforcement learning algorithm that accepts an LTL formula as task specification.

3 Related Work

Natural Language Robotic Navigation: Early work of language-guided robot task execution focused on using semantic parsers to ground language commands into abstract representations capable of

informing robot actions [31, 32, 33, 34, 35]. Recent work leverages large pretrained models to directly generate task plans, either as code [7, 6] or through text [36, 9]. The LLM output is a formal language that must be interpreted by an external procedure. Thus the external interpreters need to be expressive and competent for the success of these approaches. In contrast, planners for LTL offer asymptotic guarantees on the soundness of the resulting robot policies. Finally, LM-Nav [8] is a modular system that computes a navigational policy by using an LLM to parse landmarks from a command, then a vision-language model in conjunction with a graph search algorithm to plan over a pre-constructed semantic map. Note that LM-Nav can only ground commands of the *Sequence Visit* category as defined by Menghi et al. [28], while Lang2LTL can interpret 15 temporal tasks.

Translating Language to Formal Specification: Prior approaches are tied to the domain they were trained on, and required retraining with a large amount of data to deploy in a novel domain. Gopalan et al. [3] relied on commands paired with LTL formulas in a synthetic domain called *CleanUp World*. Patel et al. [4] and Wang et al. [5] developed a weakly supervised approach that requires natural language descriptions along with a satisfying trajectory.

Leveraging LLM to facilitate translation into LTL is an emerging research direction [37, 37, 38, 39, 40]. These approaches relied on string transformations to transform referring expressions in a command into propositions. In contrast, Lang2LTL explicitly grounds the referring expression to propositional concepts in the physical environment through a grounding module.

The closest approach to ours, proposed by Berg et al. [2], uses CopyNet [41] architecture to generate LTL formula structures followed by explicitly replacing the appropriate propositions. We demonstrate a significant performance improvement over Berg et al. [2]'s approach by leveraging LLMs and training and evaluating on a semantically diverse dataset.

4 Problem Definition

We frame the problem as users providing a natural language command u to a robotic system that performs navigational tasks in an environment $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ represent the states and actions of the robot, and T describes the transition dynamics of the environment. Our proposed language grounding system, Lang2LTL, translates the language command u to its equivalent LTL formula φ and grounds its propositions to real-world landmarks. We assume the robot has access to its state information $\mathcal{S}$ and a semantic database of propositions $\mathcal{D} = \{k : (z, f)\}$, structured as key-value pairs. The key k is a unique string identifier for each proposition, and z contains semantic information about the landmark, stored in a serialized format (e.g., JSON). For example, in a street map domain, the semantic information z includes landmark names, street addresses, amenities, etc. $f : \mathcal{S} \to \{0, 1\}$ is a Boolean valued function that evaluates the truth value of the proposition in a given state s. Appendix 2 shows an example entry of this semantic database. Finally, we assume that the robot has access to an automated planner that accepts an LTL formula and a semantic map as input and generates a plan over the semantic map as output. We used AP-MDP [12] for this paper.

Consider the example of a drone following a given command within an urban environment depicted in Figure 2. The environment $\mathcal{M}$ encodes the position and the dynamics of the drone. The semantic database $\mathcal{D} = \{k : (z, f)\}$ includes the landmark identifiers, their semantic information, and a proximity function.

5 Lang2LTL: Natural Language Grounding

Lang2LTL is a modular system that leverages LLMs to solve the grounding problem by solving the following subproblems,

1. **Referring Expression Recognition**: We identify the set of substrings, $\{r_i\}$, in the command u that refer to Boolean propositions. In this case, $\{r_i\}$ = {" the store on Main Street", "the bank"}.

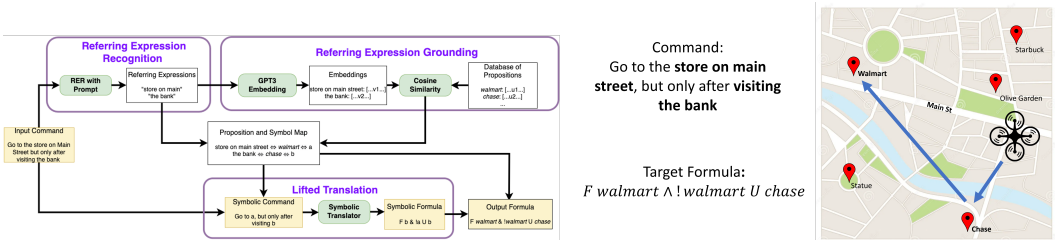


Figure 2: Lang2LTL system overview: Green blocks are pretrained or finetuned LLM models. Yellow blocks are the input or output data of each module.

2. **Referring Expression Grounding**: Each referring expression r_i is mapped to one of the proposition identifier strings, $k \in \mathcal{D}$, which yields a map $\{r_i \rightarrow k\}$. Each proposition is also bijectively mapped to a placeholder symbol β from a fixed vocabulary $\boldsymbol{\beta} = \{$"A", "B", $\ldots\}$ by $\{k \leftrightarrow \beta\}$. In the example described above, the phrases "the store on Main Street" and "the bank" refer to the proposition identifiers $Walmart$ and $Chase$, which are in turn mapped to the placeholder symbols "A" and "B".

3. **Lifted Translation**: After substituting the referring expressions $\{r_i\}$ by placeholder symbols $\boldsymbol{\beta}$ using the map $\{r_i \rightarrow \beta\}$, we obtain the lifted utterance, "Go to A, but only after visiting B.". The lifted translation module then translates this lifted utterance into a lifted LTL formula, $\varphi_{\boldsymbol{\beta}} = \mathcal{F}\,(A)\ \wedge (\neg A\,\mathcal{U}\ B)$.

Finally, we substitute placeholder symbols "A" and "B" by grounded propositions $Walmart$ and $Chase$ using the bijection $\{k \leftrightarrow \beta\}$ to construct the output LTL formula depicted in Figure 2.

We hypothesized that the benefit of solving the translation problem in the lifted domain is two-fold. First, the output vocabulary size is significantly reduced. Secondly, the lifted formula data can be sourced from multiple domains thus providing access to a larger training dataset. Once accurately translated in the lifted domain, the formulas can be grounded to unseen landmarks in novel domains, a task at which LLMs excel. We examine the efficacy of this modularization in Section 6.

5.1 Referring Expression Recognition (RER)

Referring expressions are noun phrases, pronouns, and proper names that refer to some individual objects [42]. In this work, we only consider the task of recognizing noun phrases and proper names. Referring expressions are entire substrings that refer to a single entity, therefore they are a superset of named entities. For example, "the store on Main Street" is the referring expression, but it contains two named entities, "store" and "Main Street.

Referring expression recognition is generally challenging to all existing pretrained name entity recognition models, especially without adequate examples for finetuning. We demonstrate high performance on the RER task by adapting the GPT-4 prompted with a task description and examples to enable in-context learning. Details of the prompting approach are provided in Appendix 4.

5.2 Referring Expression Grounding (REG)

Due to the diversity of natural language, a user can refer to a landmark using many possible referring expressions. Grounding these expressions into the correct propositions is challenging. We propose using the embeddings computed by an LLM for measuring similarity. LLMs have been shown to map semantically similar texts to similar embedding values [43].

Let $g_{embed} : r \rightarrow \mathbb{R}^n$ represent the function that computes an n-dimensional embedding of a text string using the parameters of the LLM. Following Berg et al. [44], we match the referring expressions $\{r_i\}$ to the proposition tokens k's by matching their respective embeddings using cosine similarity. The embedding of a proposition token, k, is computed by encoding the semantic information, z, of

its corresponding landmark, e.g., street name and amenity. This process is represented formally as follows,

$$k^* = \underset{\{k:(z,f)\}\in\mathcal{D}}{\operatorname{argmax}} \frac{g_{embed}(r_i)^\top g_{embed}(z)}{||g_{embed}(r_i)||\ ||g_{embed}(z)||} \tag{2}$$

5.3 Lifted Translation

Our lifted translation module operates with a much smaller vocabulary than the number of landmarks within any given navigation domain. It can also be trained with navigational commands from a wider variety of data sources. In designing our lifted translation module, we followed the Gopalan et al. [3], and Patel et al. [4] and represented the prediction target LTL formulas in the prefix format instead of the infix format. This allows us to unambiguously parse the formulas without requiring parenthesis matching and shorten the output.

The lifted translation module accepts a lifted utterance as an input and generates an LTL formula with an output vocabulary of up to 10 operators and five lifted propositions. We evaluated the following model classes for lifted translation. The implementation details are provided in Appendix 5.

Finetuned LLM: We finetuned LLMs using supervised learning following [16]. We tested two LLMs with supervised finetuning, namely, T5-Base (220M) [20] (using the Hugging Face Transformer library [45], and the text-davinvi-003 version of GPT-3 using the OpenAI API. The target was an exact token-wise match of the true LTL formula.

Prompt LLM: We evaluated prompting the pre-trained GPT-3 [18] and GPT-4 [19] models using the OpenAI API. We did not vary the prompts throughout a given test set.

Seq2Seq Transformers: We trained an encoder-decoder model based on the transformer architecture [15] to optimize the per-token cross entropy loss with respect to the ground-truth LTL formula, together with a token embedding layer to transform the sequence of input tokens into a sequence of high-dimensional vectors.

6 Evaluation of Language Grounding

We tested the performance of Lang2LTL towards five definitions of generalizing temporal command interpretation as informed by formal methods in Section 6.1. We evaluated the performance of each module described in Section 5 in isolation in addition to a demonstration of the integrated system. Lang2LTL achieved state-of-the-art performance in grounding diverse temporal commands in 21 novel *OpenStreetMap* regions [46] and also demonstrated the ability to ground temporal commands in indoor environments that were correctly satisfied by the Spot robot using an automated planner.

6.1 Generalization in Temporal Command Understanding

Utilizing a formal language, such as LTL to encode temporal task specifications allows us to formally define five types of generalizing behaviors.

Robustness to Paraphrasing: Consider two utterances describing the same temporal formula $\mathbf{F}\ chase$, i.e., u_1 =" Go to $chase$", and u_2 ="Visit $chase$". If the system has only seen u_1 at training time, but it correctly interprets u_2 at test time, it is said to demonstrate robustness to paraphrasing. This is the most common test of generalizations we observe in prior works on following language commands for robotics. A test-train split of the dataset is adequate for testing robustness to paraphrasing, and we refer to such test sets as *utterance holdout*. Most prior works [3, 2, 40, 37, 5, 4] demonstrate robustness to paraphrasing.

Robustness to Substitutions: Assume the system has been trained to interpret the command corresponding to $\mathbf{F}\ chase$, and $\mathbf{G}\ \neg walmart$ at training time, but has not been trained on a command corresponding to $\mathbf{F} walmart$. If the system correctly interprets the command corresponding to

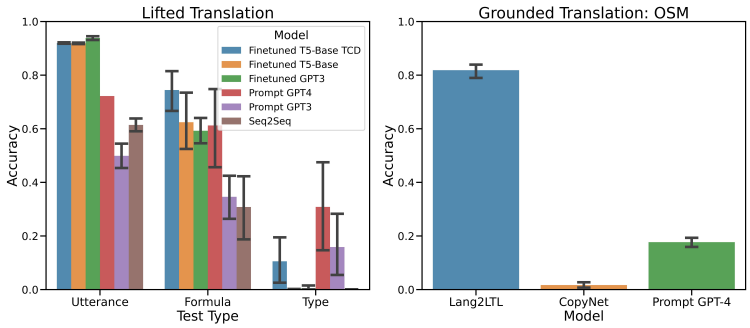
$\mathbf{F}walmart$, it is said to demonstrate robustness to substitution. To test for robustness to substitutions, any formula in the test set must not have a semantically equivalent formula in the training set. [3] demonstrated limited robustness to substitutions. The lifted translation approach followed by Berg et al. [2], NL2TL [40], and Lang2LTL demonstrates robustness to substitutions

Robustness to Vocabulary Shift: Assume the system has been trained to interpret commands corresponding to $\mathbf{F}chase$ at training time but has never seen any command containing the propositions $walmart$. If the system correctly interprets $\mathbf{F}walmart$ at test time, the system is said to be robust to vocabulary shift. To test for robustness to vocabulary shift, we identify the set of unique propositions occurring in every formula in the test and training set. The training set and test set vocabularies should have an empty intersection in addition to the non-equivalence of every test formula with respect to the training formula. Methods that explicitly rely on lifted representations show robustness to vocabulary shifts by design [44, 8, 40]. Our full system evaluations demonstrated robustness to novel vocabularies in Section 6.5.

Robustness to Unseen Formulas: Assume that all formulas in the training set are transformed by substituting the propositions in a pre-defined canonical order, e.g., both $\mathbf{F}\ chase$, and $\mathbf{F}\ walmart$ are transformed to $\mathbf{F}\ a$. We refer to these transformed formulas as the formula skeleton. To test for robustness to unseen formulas, the test set must not share any semantically equivalent formula skeleton with the training set. We used the equivalency checker built into the Spot LTL library [].

Robustness to Unseen Template Instances: We define this as a special case of robustness to unseen formulas. If all the formulas in the test and training set are derived from a template-based generator using a library of pre-defined templates [28, 47], a model may exhibit generalization to different semantically distinct formula skeletons of the same template, such a model displays robustness to unseen templates. We refer to test set whose skeletons vary only as instantiations of templates seen during training as a *formula holdout* test set. If the unseen formulas in the test set do not correspond to any patterns, we refer to it as a *type holdout* test set.

None of the prior works have evaluated proposed models for robustness to unseen formulas. We evaluated the lifted translation module of Lang2LTL on both *formula* and *type holdout* test sets. Lang2LTL experienced a degradation of performance as expected indicating that robustness to unseen formulas is still an open challenge.



(a) (b)

Figure 3: Figure 3a depicts the average accuracies of six lifted translation models on the three holdout sets over five-fold cross-validation. Figure 3b depicts the average accuracies over the grounded translation task in the *OSM* domain.

Table 1: Proposition grounding evaluation

Component	Accuracy
RE Recognition	$98.01 \pm 2.08\%$
RE Grounding	$98.20 \pm 2.30\%$

Table 2: Zero-Shot Accuracies

Dataset	Lang2LTL (ours)	Original Method
OSM [44]	$49.40 \pm 15.49\%$	45.91 ± 12.70
CleanUp [3]	$78.28 \pm 1.73\%$	$95.51 \pm 0.11\%$

6.2 Lifted Dataset

We first collected a new parallel corpus of 1,156 natural language commands in English and LTL formulas with propositional symbols to train and evaluate the lifted translation module. We included 15 LTL templates identified by Menghi et al. [28] and generated 47 unique lifted LTL formula templates by varying the number of propositions from 1 to 5 when applicable.

To improve the lifted translation module's *Robustness to Substitutions*, we permuted the propositions in utterances and corresponding formulas. The lifted dataset after permutation contains 2,125 unique LTL formulas and 49,655 English utterances describing them. For a detailed description of the dataset and data collection, please refer to Appendix 7.

6.3 Grounded Dataset

Generating Diverse REs: We used GPT-4 to paraphrase landmark names from an open-source map database, *OpenStreetMap* (*OSM*) [46], to more diverse forms of referring expressions (REs) by providing the semantic information. The prompt used for generating diverse REs is in Appendix 3.

We then substituted the symbols in the lifted dataset (Section 6.2) by diverse REs on 100 randomly sampled lifted utterances for each of the 21 *OSM* cities.

6.4 Component-wise Evaluation

Proposition Grounding: We evaluated the RER and REG modules on the grounded *OSM* dataset with 2100 utterances across 21 cities. The average accuracy and the standard error across the cities for these modules are depicted in Table 6.4

Lifted Translation: We evaluated the six models presented in Section 5.3 for lifted translation through five-fold cross-validation on *utterance*, *formula*, and *type holdout* tests. Figure 3a depicts the average accuracy and the standard deviation across the folds. We note that the two finetuned LLM models demonstrate the best performance on utterance holdout. We also noted a degradation of performance on *formula* and *type holdout* tests, with the latter being the most challenging across all models. Finetuned LLM models suffered the worst degradation of performance. Finally, the addition of task-constrained decoding to T5 significantly improved its performance on the more challenging *formula* and *type holdout*. Due to the lower cost of inference, and the ability to implement type-constrained decoding to prevent syntax errors, we chose the finetuned T5 model for lifted translation in our full-system evaluation.

6.5 System Evaluation

We compared the translation accuracy of the full Lang2LTL system on the grounded datasets with the CopyNet-based translation model [2] and Prompt GPT-4. We retrained CopyNet with an identical data budget as Lang2LTL lifted translation model. For Prompt GPT-4 we ensured that there was at least one example from each formula skeleton in the dataset included in the prompt. Figure 3b depicts Lang2LTL outperforming both the baselines by a significant margin. Note that due to the high cost of inference, Prompt GPT-4 was only evaluated on a smaller subset of the test set.

6.6 Cross-Domain Evaluation

We further tested the zero-shot generalization capability of Lang2LTL on two different crown-sourced datasets from prior work; the Cleanup World [3] on an analog indoor environment; and the *OSM* dataset [2] collected via Amazon mechanical turk. Table 2 shows the translation accuracies of Lang2LTL without any fine-tuning on the target datasets. Note that Lang2LTL outperforms CopyNet results reported by Berg et al. [2]. We further note that the CleanUp World dataset contains 6 unique formula skeletons out of which some were not a part of our lifted dataset. The degraded performance is expected when the model is needs to generalize to unseen formulas.

7 Robot Demonstration

To demonstrate Lang2LTL's ability to directly execute language commands by interfacing with automated planning algorithms, we deployed a quadruped module robot, Spot [48] with the AP-MDP planner [12] in two novel indoor environments. Each environment had eight landmarks (e.g. bookshelf, desks, couches, elevators, tables, etc.). We ensured that each environment had multiple objects of the same type but different attributes.

We used Spot's GraphNav framework to compute a semantic map of the environment. The AP-MDP [12] planner can directly plan over this semantic map, given an LTL task specification. Please refer to Appendix 10 for a complete description of the task environments.

As a proof-of-concept, we further finetuned the lifted translation module on 120,000 lifted utterances and formulas formed by sampling pairwise compositions from the lifted database and composed using conjunctions and disjunctions.

We compared Lang2LTL to Code-as-policies CaP [6] a prominent example of directly grounding language instructions to robot plans expressed as Python code. We provided Code-as-policies with interface functions to input the semantic map, and a helper function to automatically navigate between nodes while having the ability to avoid certain regions. Thus CaP had access to actions at a much higher level of abstraction than our system. Note that AP-MDP only interfaced with primitive actions defined as movement between any two neighboring nodes.

Lang2LTL was able to correctly translate 52 commands (40 satisfiable commands, and 12 unsatisfiable commands), the formal guarantees of the AP-MDP planner assured that the robot execution was aborted when facing an unsatisfiable specification. By contrast, CaP was only able to generate acceptable executions for 23 out of 52 commands and did not explicitly recognize the unsatisfiable commands. We do note that CaP demonstrated more robustness to paraphrasing than our system which failed on some compositional patterns not included in the augmented lifted dataset.

8 Limitations

We observed that Lang2LTL fails at grounding language commands with certain utterance structures, which suggests that the lifted translation model overfits some training utterances. A list of incorrect groundings is shown in Appendix Table 1 and Table 2. Finetuning models with larger capacities, e.g., T5-Large, may help. If there are multiple landmarks of the same semantic features present in the environment, e.g., two Starbucks, Lang2LTL cannot distinguish the two and selects one at random. To resolve this ambiguity, the robot needs to actively query the human user via dialog when necessary.

9 Conclusion

We propose Lang2LTL, a modular system using large language models to ground complex navigational commands for temporal tasks in novel environments of household and city scale without re-training, and generalization tests for language grounding systems. Lang2LTL achieves a grounding accuracy of 81.83% in 21 unseen cities and outperforms the previous SoTA and an end-to-end prompt GPT-4 baseline. Any robotic system equipped with position-tracking capability and a semantic map with landmarks labeled with free-form text can utilize Lang2LTL to interpret natural language from human users without additional training.

References

[1] A. Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57. ieee, 1977.

[2] M. Berg, D. Bayazit, R. Mathew, A. Rotter-Aboyoun, E. Pavlick, and S. Tellex. Grounding Language to Landmarks in Arbitrary Outdoor Environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[3] N. Gopalan, D. Arumugam, L. Wong, and S. Tellex. Sequence-to-Sequence Language Grounding of Non-Markovian Task Specifications. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, 2018. doi:10.15607/RSS.2018.XIV.067.

[4] R. Patel, E. Pavlick, and S. Tellex. Grounding language to non-markovian tasks with no supervision of task specifications. In *Robotics: Science and Systems*, 2020.

[5] 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*, pages 1706–1718. PMLR, 2021.

[6] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. In *arXiv preprint arXiv:2209.07753*, 2022.

[7] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg. Progprompt: Generating situated robot task plans using large language models. *arXiv preprint arXiv:2209.11302*, 2022.

[8] D. Shah, B. Osiński, S. Levine, et al. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. In *Conference on Robot Learning*, pages 492–504. PMLR, 2022.

[9] S. S. Raman, V. Cohen, E. Rosen, I. Idrees, D. Paulius, and S. Tellex. Planning with large language models via corrective re-prompting. *arXiv preprint arXiv:2211.09935*, 2022.

[10] M. L. Littman, U. Topcu, J. Fu, C. Isbell, M. Wen, and J. MacGlashan. Environment-independent task specifications via gltl. *arXiv preprint arXiv:1704.04341*, 2017.

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

[12] Y. Oh, R. Patel, T. Nguyen, B. Huang, E. Pavlick, and S. Tellex. Planning with State Abstractions for Non-Markovian Task Specifications. In *Proceedings of Robotics: Science and Systems*, Freiburg, Germany, June 2019.

[13] J. X. Liu, A. Shah, E. Rosen, G. Konidaris, and S. Tellex. Skill transfer for temporally-extended task specifications. *arXiv preprint arXiv:2206.05096*, 2022.

[14] R. T. Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research*, 73:173–208, 2022.

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

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

[17] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[18] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[19] OpenAI. Gpt-4 technical report, 2023.

[20] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

[21] R. T. Icarte, T. Klassen, R. Valenzano, and S. McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*, pages 2107–2116. PMLR, 2018.

[22] G. De Giacomo, L. Iocchi, M. Favorito, and F. Patrizi. Foundations for restraining bolts: Reinforcement learning with ltlf/ldlf restraining specifications. In *Proceedings of the international conference on automated planning and scheduling*, volume 29, pages 128–136, 2019.

[23] A. Shah, S. Li, and J. Shah. Planning with uncertain specifications (puns). *IEEE Robotics and Automation Letters*, 5(2):3414–3421, 2020.

[24] A. Shah, P. Kamath, J. A. Shah, and S. Li. Bayesian inference of temporal task specifications from demonstrations. *Advances in Neural Information Processing Systems*, 31, 2018.

[25] M. Vazquez-Chanlatte, S. Jha, A. Tiwari, M. K. Ho, and S. Seshia. Learning task specifications from demonstrations. In *Advances in Neural Information Processing Systems 31*, pages 5368–5378. 2018.

[26] J. Kim, C. Muise, A. Shah, S. Agarwal, and J. Shah. Bayesian inference of linear temporal logic specifications for contrastive explanations. In *IJCAI*, 2019.

[27] A. Shah, S. Wadhwania, and J. Shah. Interactive robot training for non-markov tasks. *arXiv preprint arXiv:2003.02232*, 2020.

[28] C. Menghi, C. Tsigkanos, P. Pelliccione, C. Ghezzi, and T. Berger. Specification patterns for robotic missions. *IEEE Transactions on Software Engineering*, 47(10):2208–2224, oct 2021. ISSN 1939-3520. doi:10.1109/TSE.2019.2945329.

[29] M. Y. Vardi. An automata-theoretic approach to linear temporal logic. *Logics for concurrency*, pages 238–266, 1996.

[30] R. Gerth, D. Peled, M. Y. Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Protocol Specification, Testing and Verification XV: Proceedings of the Fifteenth IFIP WG6. 1 International Symposium on Protocol Specification, Testing and Verification, Warsaw, Poland, June 1995*, pages 3–18. Springer, 1996.

[31] M. Macmahon. Marco: A modular architecture for following route instructions. *AAAI Workshop - Technical Report*, 01 2005.

[32] M. MacMahon, B. Stankiewicz, and B. Kuipers. Walk the talk: Connecting language, knowledge, and action in route instructions. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, page 1475–1482. AAAI Press, 2006. ISBN 9781577352815.

[33] D. L. Chen and R. J. Mooney. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI'11, page 859–865. AAAI Press, 2011.

[34] J. Kim and R. Mooney. Unsupervised PCFG induction for grounded language learning with highly ambiguous supervision. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 433–444, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL https://aclanthology.org/D12-1040.

[35] Y. Artzi and L. Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62, 2013. doi:10.1162/tacl_a_00209. URL https://aclanthology.org/Q13-1005.

[36] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[37] J. Pan, G. Chou, and D. Berenson. Data-efficient learning of natural language to linear temporal logic translators for robot task specification. *arXiv preprint arXiv:2303.08006*, 2023.

[38] M. Cosler, C. Hahn, D. Mendoza, F. Schmitt, and C. Trippel. nl2spec: Interactively translating unstructured natural language to temporal logics with large language models. *arXiv preprint arXiv:2303.04864*, 2023.

[39] F. Fuggitti and T. Chakraborti. NL2LTL – a python package for converting natural language (NL) instructions to linear temporal logic (LTL) formulas. In *AAAI*, 2023. System Demonstration.

[40] 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.

[41] J. Gu, Z. Lu, H. Li, and V. O. Li. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi:10.18653/v1/P16-1154. URL https://aclanthology.org/P16-1154.

[42] J. Lyons. *Semantics: Volume 2*, volume 2. Cambridge university press, 1977.

[43] A. Neelakantan, T. Xu, R. Puri, A. Radford, J. M. Han, J. Tworek, Q. Yuan, N. Tezak, J. W. Kim, C. Hallacy, et al. Text and code embeddings by contrastive pre-training. *arXiv preprint arXiv:2201.10005*, 2022.

[44] M. Berg, D. Bayazit, R. Mathew, A. Rotter-Aboyoun, E. Pavlick, and S. Tellex. Grounding Language to Landmarks in Arbitrary Outdoor Environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[45] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, Oct. 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[46] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2017.

[47] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett. Patterns in property specifications for finite-state verification. In *Proceedings of the 21st international conference on Software engineering*, pages 411–420, 1999.

[48] Boston Dynamics. Spot® - the agile mobile robot. https://www.bostondynamics.com/products/spot.