Robotic Language Grounding

By

Jason Xinyu Liu

A dissertation submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in the Department of Computer Science at Brown University

Providence, Rhode Island

October 2025

This dissertation by Jason Xinyu Liu is accepted in its present form by the Department of Computer Science as satisfying the dissertation requirement for the degree of Doctor of Philosophy.

Date _____ _____

Stefanie A. Tellex, Advisor

Recommended to the Graduate Council

Date _____ _____

George D. Konidaris, Reader

Date _____ _____

Michael L. Littman, Reader

Date _____ _____

Ellie Pavlick, Reader

Date _____ _____

Julie Shah, Reader

Approved by the Graduate Council

Date _____ _____

Janet A. Blume, Dean of the Graduate School

Curriculum Vitae

Education

2021–2025 Ph.D., Computer Science
Brown University (Providence, RI, USA)

2018-2021 M.S., Computer Science
Brown University (Providence, RI, USA)

2014–2017 B.S., Electrical Engineering and Computer Sciences
University of California, Berkeley (Berkeley, CA, USA)

Publications

Jia M., Huang H., Zhang Z., Wang C., Zhao L., Wang D., **Liu J.X.**, Walters R., Platt R., Tellex S. (2025). Open-vocabulary Pick and Place via Patch-level Semantic Maps. *IEEE Robotics and Automation Letters (RA-L).*

Ying L., Liu, **Liu J.X.**, Aarya S., Fang Y., Tellex S., Tenenbaum J., B; Shu T. (2024). SIFToM: Robust Spoken Instruction Following through Theory of Mind. *arXiv preprint arXiv:2409.10849.*

Liu J.X., Shah A., Konidaris G.D., Tellex S., Paulius and D. (2024). Lang2LTL-2: Grounding Spatiotemporal Navigation Commands Using Large Language and Vision-Language Models. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*

Cohen V.*, **Liu J.X.***, Mooney R.*, Tellex S.*, Watkins D.* (2024). A Survey of Robotic Language Grounding: Tradeoffs between Symbols and Embeddings. *International Joint Conference on Artificial Intelligence (IJCAI) Survey Track.*

Liu J.X.*, Shah A.*, Rosen E., Jia M., Konidaris G., Tellex S. (2024). LTL-Transfer: Skill Transfer for Temporally-Extended Task Specifications. *IEEE International Conference on Robotics and Automation (ICRA).*

Hori C., Peng P., Harwath D., **Liu J.X.**, Ota K., Jain S., Corcodel R., Jha D., Romeres D., Le Roux J. (2023). Style-transfer based Speech and Audio-visual Scene Understanding for Robot Action Sequence Acquisition from Videos. *INTERSPEECH 2023.*

Liu J.X.*, Yang Z.*, Idrees I., Liang S., Schornstein B., Tellex S., Shah A. (2023) Grounding Complex Natural Language Commands for Temporal Tasks in Unseen Environments. *Conference on Robot Learning (CoRL).*

Hsiung E., Mehta H., Chu J., **Liu J.X.**, Patel R., Tellex S., Konidaris G. (2022). Generalizing to New Domains by Mapping Natural Language to Lifted LTL. *IEEE*

International Conference on Robotics and Automation (ICRA).

McCoyd M., Park W., Chen S., Shah N., Roggenkemper R., Hwang M., **Liu J.X.**, Wagner D. (2020). Minority Reports Defense: Defending Against Adversarial Patches. *Applied Cryptography and Network Security Workshops (ACNS).*

Mahler J., Matl M., **Liu J.X.**, Li A., Gealy D., Goldberg K. (2018) Dex-Net 3.0: Computing Robust Vacuum Suction Grasp Targets in Point Clouds Using a New Analytic Model and Deep Learning. *IEEE International Conference on Robotics and Automation (ICRA).*

J.X. Liu, Wagner D., Egelman S. (2018). Detecting Phone Theft Using Machine Learning. *International Conference on Information Science and System (ICISS).*

Mahler J., Liang J., Niyaz S., Laskey M., Doan R., **Liu J.X.**, Ojea J. A., Goldberg K. (2017). Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics. *Robotics: Science and Systems (RSS).*

Invited Talks

December 6, 2024, **Brown Database Group**
Brown University (Providence, RI, USA)

October 25, 2024, **Amazon Robotics Research Fall Symposium and PhD Communication Competition**
Amazon Robotics (Cambridge, MA, USA)

October 11, 2024, **Forum for Artificial Intelligence**
The University of Texas at Austin (Austin, TX, USA)

October 3, 2024, **CS Colloquium**
University of Colorado Boulder (Boulder, CO, USA)

September 25, 2024, **Interactive Robotics Group**
Massachusetts Institute of Technology (Cambridge, MA, USA)

September 13 2024, **Center of Autonomy**
The University of Texas at Austin (Austin, TX, USA)

August 12, 2024, **Logos Robotics Lab**
Arizona State University (Tempe, AZ, USA)

July 12, 2024, **CS Department Seminar**
Oxford University (Oxford, UK)

October 8, 2022, **Northeast Robotics Colloquium**
UMass Lowell (Lowell, MA, USA)

Invited Lectures

Spring 2025, **CSCI 1470: Deep Learning**
Brown University (Providence, RI, USA)

Fall 2024, **CSCI 1951X: Formal Proof and Verification**
Brown University (Providence, RI, USA)

Awards and Fellowships

Jack Kent Cooke Foundation Graduate Scholarship 2018

National Science Foundation Graduate Research Fellowship Program 2018

Service

Reviewer: Journal of Machine Learning Research (JMLR) 2025

Reviewer: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2024

Reviewer: IEEE International Conference on Robotics and Automation (ICRA) 2023

Reviewer: IEEE International Conference on Robotics and Automation (ICRA) 2022

Mentor: Brown First-year Ph.D. student mentor 2021-2023

Mentor: Brown Google exploreCSR Spring 2021

Acknowledgments

I would like to first express my gratitude to my advisors and mentors. Prof. Stefanie Tellex took a chance on me and gave me the opportunity to pursue what I love to do, research in robotics. Through many wonderful years, Stefanie has given me all the support I need to become an independent researcher. She also taught me how to lead with humanity and compassion. I hope I will continue working on cool projects that "make robots do something they cannot do before," as Stefanie always says. Prof. George Konidaris is not my official advisor, but he has always been there for me when I need his help and wisdom. During the time we worked together, I learned a great deal of strategic planning for my research and career, in addition to technical skills, from George, usually in his well-lit office with a good smile of tea or Dr. Pepper. Prof. Michael Littman taught me how to approach research with scientific rigor and fun. I took Michael's Sequential Decision Making class, which was filled with rigorous mathematical derivation and laughter. Michael also gave me one of the most important advice for my career, "do not think it is something that is out of reach." When I first started my Ph.D. doing research on RoboNLP in 2018, approximately four GPTs ago, I needed to learn natural language processing (NLP), a field that was brand new to me. Prof. Ellie Pavlick taught me a great deal of the fundamentals and important problems in NLP, which continues serving as inspiration for my research. After meeting Prof. Julie Shah and many incredible graduates from her lab, I knew it was the right place for my postdoc because of our well-aligned research interests and the amazing lab culture she helps cultivate. I deeply appreciate that Julie and her students

welcomed me with open arms. I am forever grateful to my undergraduate advisors, Prof. Ken Goldberg and Prof. David Wagner. Without their guidance and encouragement, my journey into research and my pursuit of a Ph.D. degree would not have been possible. I am also grateful to my mentors, David Abel, Cameron Allen, Jeff Mahler, Nathan Malkin, Austin Murdock, Nakul Gopalan, Tiffany Reardon, and Ankit Shah. They have always made time to listen and offer their advice.

My parents, Hong Zhou and Weidong Liu, decided to uproot and relocate to the United States. They gave up their comfortable lives in China, so I could have more opportunities. As a first-generation college student, I would not be where I am today without their unwavering support and love. I love you, Mom and Dad. I hope I have made you proud.

The Brown Computer Science Department cultivates an exceptionally collaborative environment. I had the honor and privilege to learn from many faculty members. I appreciate your time and guidance: Nora Ayanian, Stephen Bach, Serena Booth, Ugur Cetintemel, Robert Lewis, Jeff Huang, Daniel Ritchie, Srinath Sridhar, James Tompkin, Eli Upfal.

To my collaborators and friends who supported me during my undergraduate and graduate studies, I cherish the time we spent together and the beautiful memories we created: Ben Abbatematteo, Alper Ahmetoglu, Tuluhan Akbulut, Akhil Bagaria, Joey Bai, Matthew Berg, Jonathan Chang, Yongchao Chen, Catherine Chen, Matthew Corsaro, Nicholas DeMarinis, Eric Ewing, Michael Fishman, Jessica Forde, Haotian Fu, Julia Fu, Aditya Ganeshan, Dan Haramati, Eric Hsiung, Richard Huang, Ifrah Idrees, Mingxi Jia, Yeganeh Kordi, Hongyu Li, Jasmine Liu, Charles Lovering, Sam Lobel, Rachael Ma, Max Merlin, Nihal Nayak, Thao Nguyen, Shane Parr, David Paulius, Lucy Qin, Rafael Rodriguez-Sanchez, Eric Rosen, Sam Saarinen, Naman Shah, Seiji Shaw, Saket Tiwari, Aaron Traylor, Shivam Vats, Felix Wang, Guixing Wei (the man who taught me how to dress), David Whitney, Brandon Woodard, Zhongxia Yan, Albert Yu, Peilin Yu, Tian Yun, Zilai Zeng, George Zerveas, Jeffery Zhang, Yuxuan Zhao, Kaiyu Zheng, Max Zuo,

and the members from H2R, IRL, RLAB, Brown CS, and Brown Graduate School.

Lastly, I would like to express my sincere gratitude to the Jack Kent Cooke Foundation and the National Science Foundation for their financial support throughout my undergraduate and graduate studies. I hope one day I can give back to others like they have helped me.

Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

Robots are becoming capable and prevalent in human environments. An ever-growing number of autonomous vehicles are on the road, driving next to human drivers. Robots support workers in fulfillment centers to package customer orders, and healthcare providers in hospitals by collecting and delivering medical supplies. It is crucial to specify tasks accurately so that robot execution is effective and safe.

There are many interfaces for commanding robots. Virtual reality and touchscreens are intuitive and expressive, but need users to specify the desired robot pose. Computer programs are expressive and unambiguous, but require expertise to write. Natural language provides an intuitive, expressive, and flexible way for humans to communicate with robots. Human beings have evolved to use language for communication [1, 2]. We acquire words and meanings since infancy [3], so we can use natural language fluently to talk to each other. When we have personal assistant robots in millions of homes, the most natural way for us to communicate with our robots is through natural language. With advances in large language models (LLMs) and vision language models (VLMs), we would want to talk to robots like talking to other people.

Developing intelligent machines that can communicate with people using natural

language, the language humans use to communicate, has sparked interests since the inception of artificial intelligence [4, 5]. However, grounding complex and ambiguous language in the physical world is challenging. The problem of **Robotic Language Grounding** addresses the challenge of connecting the words in a natural language utterance to the corresponding robot perception and actions in the physical world. For example, as shown in Figure 1.1, given the natural language command "Hey robot, can you fetch me the cup?" the robot must associate the word "cup" with the green cylindrical object in front of it and the word "fetch" with the motion of grasping the "cup" and delivering it to the user. We need to answer three questions in developing a robot system

Figure 1.1: An example of robotic language grounding where the robot must associate the world "cup" with the green cylindrical object in front of it, and the word "fetch" with the motion of grasping the "cup" and delivering it to the user to complete the natural language command "Hey robot, can you fetch me the cup?"

that can communicate with people using natural language.

1. What grounding representation to use?

2. How to ground language to the representation of choice?

3. How to produce robot actions from the grounding representation?

Recent works use LLMs and VLMs to ground natural language to various symbolic representations and high-dimensional embeddings, but they do not generalize to semantically diverse commands in novel environments. In addition, when performing new tasks,

robots often have to learn from scratch. This thesis introduces methods to ground natural language from humans to a structured representation in novel environments without retraining on language data, then uses the structure to induce robot actions to complete novel tasks zero-shot. In this dissertation, we demonstrate the language grounding system in novel cities and deploy it at the task planning level of a real robot in indoor and outdoor environments to solve complex navigation and mobile manipulation tasks that require the robot to understand semantically diverse temporal and spatiotemporal commands.

Figure 1.2 shows an overview of the three main components of this dissertation that jointly answer the three questions mentioned above to tackle the problem of robotic language grounding. We conclude by explaining the limitations of our approaches and proposing future research directions towards Bidirectional Multimodal Human-Robot Interaction.

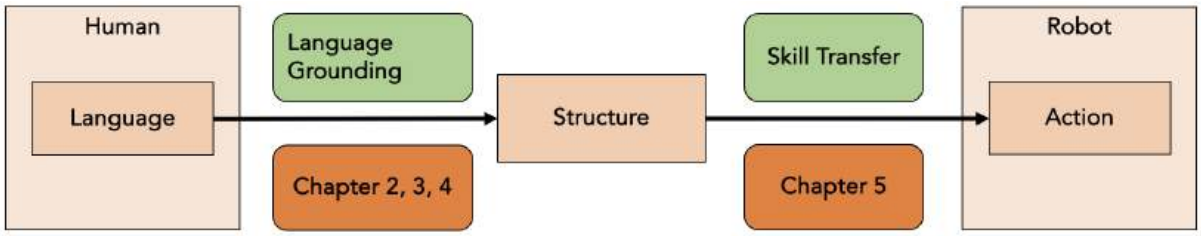


Figure 1.2: An overview of this dissertation that summarizes a robotic system that grounds natural language from human users to a structure, then uses the structure to produce robot actions.

Robotic Language Grounding: We have motivated that natural language is an effective way to convey task specifications to robots, and the problem of robotic language grounding is essential to effective and safe robot execution. However, we lack a systematic framework that governs work that grounds language to different representations in the field of robotics. To address this issue, in Chapter 2, we review and situate recent literature into a spectrum with two poles: 1) mapping between language and some manually defined formal representation of meaning, and 2) mapping between language and high-dimensional vector spaces that translate directly to low-level robot policy, as shown in Figure 2.1. We then propose some desired properties of an effective grounding representation, which we

use to guide the discussion of the tradeoffs among various grounding representations. Using a formal representation allows the meaning of the language to be precisely represented, limits the size of the learning problem, and leads to a framework for interpretability and formal safety guarantees. Methods that embed language and perceptual data into high-dimensional spaces avoid this manually specified symbolic structure and thus have the potential to be more general when fed enough data, but require more data and computing to train. In this dissertation, we will investigate the grounding of natural language in a structured representation, Linear Temporal Logic, because it can capture diverse spatiotemporal constraints and provide safety guarantees. We will also leverage its compositionality to reuse learned skills to solve novel tasks zero-shot.

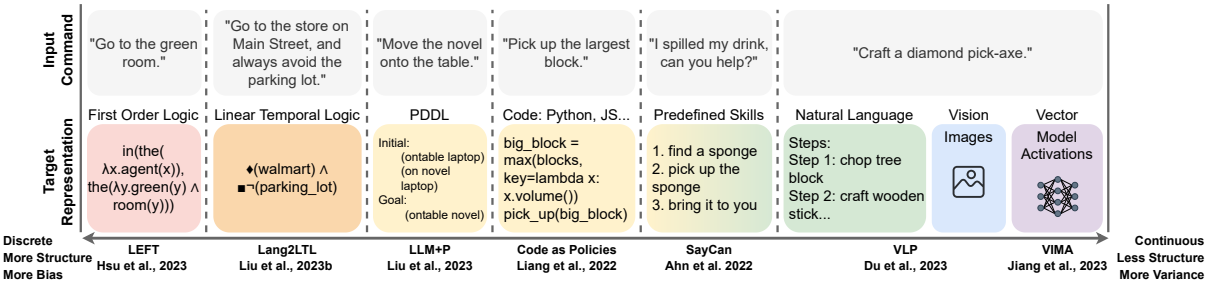


Figure 1.3: Approaches to representing natural language for robotics fall along a spectrum from more symbol-like representations to more continuous embedding-like representations. However, most approaches use a mixture of both.

Grounding Temporal Language: 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. In Chapter 3, we propose Lang2LTL, a modular system and a software package that leverages large language models (LLMs) to ground temporal navigational commands to LTL task 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 that can follow 52 semantically diverse navigational commands in two indoor environments.

Figure 1.4: Lang2LTL can ground complex navigational commands in household and city-scaled environments without retraining.

Grounding Spatiotemporal Language: Grounding spatiotemporal navigation commands to structured task specifications enables autonomous robots to understand a broad range of natural language and solve long-horizon tasks with safety guarantees. Prior works mostly focus on grounding spatial or temporally extended language for robots. In Chapter 4, we propose Lang2LTL-2, a modular system that leverages pretrained large language and vision-language models and multimodal semantic information to ground spatiotemporal navigation commands in novel city-scaled environments without retraining. Lang2LTL-2 achieves 93.53% language grounding accuracy on a dataset of 21,780 semantically diverse natural language commands in unseen environments. We run an ablation study to validate the need for different modalities. We also show that a physical robot equipped with the same system without modification can execute 50 semantically diverse natural language commands in both indoor and outdoor environments.

Skill Transfer Using Structured Task Representation: Deploying robots in real-world environments, such as households and manufacturing lines, requires generalization across novel task specifications without violating safety constraints. Linear temporal logic (LTL) is a widely used task specification language with a compositional grammar that naturally induces commonalities among tasks while preserving safety guarantees. However, most prior work on reinforcement learning with LTL specifications treats every

Figure 1.5: Lang2LTL-2 grounds spatiotemporal navigation commands in indoor and outdoor environments. The spatial and temporal components of the example commands are highlighted in blue and red, respectively.

new task independently, thus requiring large amounts of training data to generalize. We propose LTL-Transfer, a zero-shot transfer algorithm that composes task-agnostic skills learned during training to safely satisfy a wide variety of novel LTL task specifications. Experiments in Minecraft-inspired domains show that after training on only 50 tasks, LTL-Transfer can solve over 90% of 100 challenging unseen tasks and 100% of 300 commonly used novel tasks without violating any safety constraints. We deployed LTL-Transfer at the task-planning level of a quadruped mobile manipulator to demonstrate its zero-shot transfer ability for fetch-and-deliver and navigation tasks.

(a) go to shelf to fetch book (b) deliver juice to desk

Figure 1.6: The robot is executing two of four pretrained task-agnostic skills sequentially to solve a novel task $\mathbf{F}(book \wedge \mathbf{F}(desk_a \wedge \mathbf{F}(juice \wedge \mathbf{F}desk_a)))$, i.e., fetch and deliver a book then a juice bottle to the user.

CHAPTER 2

Robotic Language Grounding

In this dissertation, we use natural language, the language humans use to communicate, as an intuitive, expressive, and flexible way for human-robot communication. With advances in large language models (LLMs) and vision language models (VLMs), we should be able to talk to robots like talking to other people. However, language commands from humans can be complex and ambiguous. The problem of **Robotic Language Grounding** addresses the challenge of connecting the words in a natural language utterance to the corresponding robot perception and actions in the physical world. In this chapter, we propose a framework that situates work on grounding language onto different representations in the field of robotics. We review and situate literature into a spectrum with two poles: 1) mapping between language and some manually defined formal representation of meaning, and 2) mapping between language and high-dimensional vector spaces that translate directly to low-level robot policy. We propose some desired properties of an effective grounding representation and use the spectrum to guide our discussion of the tradeoffs among various grounding representations. Using a formal representation allows the meaning of the language to be precisely represented, limits the size of the learning problem, and leads to interpretability and formal safety guarantees. Methods that embed language and perceptual data into high-dimensional spaces avoid

this manually specified symbolic structure and thus have the potential to be more general when fed enough data, but require more data and computing to train.

2.1 Introduction

Large language models (LLMs) have fueled a surge of interest in the problem of making robots understand natural language commands. Solving this problem requires mapping between words in language and actions or behaviors taken by the robot. [6, 7] defined the symbol grounding problem as constructing a mapping from symbols of a symbolic system or words in an utterance to sensorimotor substrates in the physical world. Large language models (LLMs) semantically represent concepts without explicit higher-order symbols beyond the words in the text, leading many to try end-to-end approaches for robotic language understanding. Yet many recent works in robotic language understanding leverage large language models hand-in-hand with formal symbolic representations. For example, Code as Policies [8] generates Python code with predefined Python APIs. SayCan [9] grounds natural language commands to predefined discrete skills implemented using a deep neural network. Other approaches take a more end-to-end approach, such as VIMA [10], which learns a mapping from vision and language instructions to low-level robot actions such as joint states.

This survey paper evaluates work that grounds natural language to robot behavior. We observe that these approaches can be situated on a spectrum ranging between two high-level approaches: mapping between language and a manually defined formal representation and mapping between language and high-dimensional vector spaces that translate directly to low-level robot policy. We define the advantages and limitations of each approach.

Using a formal representation constrains the search space during training and inference and may require less training data. It also provides a natural framework for strong interpretability and formal safety guarantees. Well-defined model-checking tools exist to check

whether the model of a system meets a given logical specification [11]. Formal methods can also synthesize correct-by-construction robot controllers given a logical specification and the system model and provide counterexamples to explain failure cases [12]. However, a formal representation constrains the space of possible models that can be learned, limiting the system's ability to represent the meanings of what a person may say. With the advent of large language models, mapping between human language and a formal language is much easier; the research questions then need to focus on what formal language to use, where it comes from, and how it connects to the physical world. There are opportunities to more easily use existing representations such as the planning domain definition language (PDDL), linear temporal logic (LTL), or motion planners without collecting large training sets. Many approaches at the border, such as SayCan [9], map language to a manually specified vocabulary of robot skills but give their formal language relatively little attention despite it playing a critical role in the system.

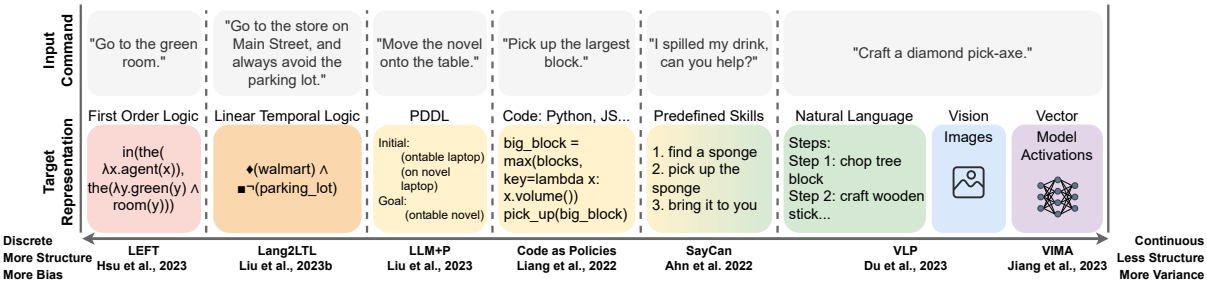


Figure 2.1: Approaches to representing natural language for robotics fall along a spectrum from more symbol-like representations to more continuous embedding-like representations. However, most approaches use a mixture of both. SayCan uses a fixed ontology of predefined skills but implements these as neural value functions conditioned on language.

End-to-end approaches may require more data to train but are more flexible in representing a user's intended meaning and translating it to robot behavior. [13] observed that structured generative models perform better with less training data. In contrast, discriminative models with more parameters can perform better when given lots of data because they make fewer assumptions about the structure of the learned model. Similarly, end-to-end neural embedding approaches require more data but can generalize better than formal methods because they place fewer constraints on the learned model. True

"pixels to torque" approaches [14] learn to produce motor torques directly from sensor input; however, many end-to-end approaches use intermediate outputs such as end-effector poses or joint states. The challenge with end-to-end approaches is acquiring enough training data, processing this data, generalizing outside the training set, chaining policies together to produce long-term behaviors, explaining robot behaviors, and providing safety guarantees.

We situate the robotic language grounding works on a spectrum in Figure 2.1 and describe more formal work in Section 2.2 and more end-to-end work in Section 2.3. More formal work uses discrete and structured representations that introduce bias into the learning. We begin with the most structured, abstract representations and successively review less structured, lower-level, continuous representations. Many of the implementations surveyed combine aspects of formal and end-to-end representations. In Section 2.4, we survey output representations for both approaches, review available datasets, and discuss the scope of natural language commands understood in theory and practice by different approaches. We conclude by identifying key open problems and recommending future research directions.

This new review paper focuses on the transformative role LLMs have played in this space. Other relevant reviews are:

- Tellex et al. [15] survey robot and language grounding work predating LLMs' emergence.
- Zhang et al. [16] review large language models more broadly used in human-robot interaction, including question answering, social robotics, and instruction following.
- Zeng et al. [17] review LLMs applied to robotics broadly, including related technologies, but does not focus on the spectrum from formal methods to high-dimensional vectors as this paper does.
- Wang et al. [18] review applications of LLMs to robotics but does not situate the work

on a spectrum and focuses on a broader set of tasks than command understanding.

Our work, in contrast, focuses specifically on the problem of command understanding, situating work along a spectrum based on formal methods.

	Formal vs. Intermediate vs. End-to-End	Intermediate Grounding	Final System Output	Executing Output on Robots	Domains
LEFT [19]	Formal	N/A	FOL formula	Low-level controllers	Pick-and-Place (Real)
Lang2LTL [20]	Formal	N/A	LTL formula	LTL-MDP planner, Low-level controllers	Navigation (Real)
[21]	Formal	N/A	PDDL goal	Symbolic planner, Low-level controllers	Navigation, Mobile manipulation (Sim)
LLM+P [22]	Formal	N/A	PDDL	Symbolic planner, Low-level controllers	Manipulation (Real)
SayCan [9]	Formal	N/A	predefined skill	Low-level controllers	Mobile manipulation (Real)
Code as Policies [8]	Formal	N/A	Python code	Python interpreter, Low-level controllers	Pick-and-place, Mobile manipulation Drawing (Real)
ProgPrompt [23]	Formal	N/A	Python code	Python interpreter, Low-level controllers	Mobile manipulation (Sim) Manipulation (Real)
Voyager [24]	Formal	N/A	Javascript Code	Javascript interpreter, Low-level controllers	Minecraft survival tasks (Sim)
ITP [25]	Intermediate	High-level action descriptions	Python code	Python interpreter, Low-level controllers	Manipulation (Real)
UniSim [26]	Intermediate	Image subgoals	predefined skill	Low-level controllers	Manipulation (Real)
SuSIE [27]	Intermediate	Image subgoals	End-effector pose	Low-level controllers	Manipulation (Real)
VLP [28]	Intermediate	Image subgoals	End-effector pose	Low-level controllers	Manipulation (Real)
PaLM-E [29]	End-to-End	N/A	predefined skill	Low-level controllers	Pick-and-place, Manipulation (Real)
VIMA [10]	End-to-End	N/A	predefined skill	Low-level controllers	Manipulation (Sim)
PerAct [30]	End-to-End	N/A	End-effector pose	Low-level controllers	Manipulation (Real)
RT-1 [31]	End-to-End	N/A	End-effector pose	Low-level controllers	Manipulation (Sim, Real) Mobile Manipulation (Real)
RT-2 [32]	End-to-End	N/A	End-effector pose	Low-level controllers	Manipulation (Real)
RT-X [33]	End-to-End	N/A	End-effector pose	Low-level controllers	Manipulation (Real)

Table 2.1: Method Comparison

2.2 Mapping from Natural Language to a Formal Representation

Works closer to the formal end of the spectrum map natural language commands from humans to a manually defined formal representation, e.g., temporal logic, planning domain definition language (PDDL), computer code, or some predefined skills. The symbols in the formal representation are then grounded to robot percepts and control by predefined detectors and controllers, respectively. Unlike machine translation of natural languages, where vast training data is available online, translating natural language to logic often lacks labeled pairs of natural language commands and logic formulas. Most recent works leverage few-shot learning or fine-tuning of large language models (LLMs) for various parts of the language grounding system to address the lack of training data.

Many formal representations used to ground natural language are Turing-complete and

thus can be translated from one to another. However, depending on the language used, this translation may be direct or indirect and require significantly longer or more complex expressions. For example, linear temporal logic (LTL) can naturally represent English sentences such as "Avoid the red room" with a short, direct expression. Python can represent the same command by defining an "avoid" function but may need a significantly longer program if the function is not provided. Thus, in our review, we order these representations from those with more structure and bias to those with less structure and less bias, as typically used, shown in Figure 2.1.

More structured methods often map directly to certain natural language commands and express the goal or constraint directly rather than imperatively how to achieve it. Goal-based representations specify what state the world should be in but not what actions the robot should take to attain that state. In contrast, action-based representations specify a sequence of actions but not necessarily the goal or results of those actions. A goal-based representation for "avoid the red room" might be a logical formula such as $\neg red\_room$ while an action-based expression might be $North; North; West; West; North; North$ (depending on the specific geometry of the environment). We order our review from high-level, abstract representations to low-level, concrete, fine-grained representations.

2.2.1 Logics

Logics are mathematically precise goal-based representations that specify robotic goals and provide guarantees for robot behaviors. Temporal logics can concisely represent long-horizon, temporally extended tasks. [12] surveyed the uses of several temporal logics as task specifications for the formal synthesis of robot controllers. We consider logical expressions at the most formal end of the spectrum because they map to goals, describing abstractly the state of the world corresponding to the language, leaving the plan to achieve this state of the world to other modules.

To train their language grounding system on diverse natural language commands, [34]

used LLMs to paraphrase structured English commands generated from algorithmically produced LTL formulas. [35] and [36] trained a semantic parser to map language commands to LTL formulas using weak supervision of execution trajectories without any LTL annotations. Lang2LTL [20] is a modular system that uses LLMs to ground navigation commands to linear temporal logic (LTL) formulas and their propositions to physical landmarks in a given semantic map. The same system solved navigational tasks in indoor and outdoor environments without retraining on language data by harnessing pre-trained LLMs. Similarly, [37] first translated commands to lifted LTL formulas and then grounded them to specific domains for better generalization. Its extension Lang2LTL-2 [38] grounds spatiotemporal natural language commands with an additional spatial predicate grounding module that uses predefined rules for grounding spatial relations. Other approaches [39, 40] also used LLMs to translate natural language commands to logical representations but did not ground the formulas to a robot domain.

AutoTAMP [41] uses LLMs to translate task and state descriptions to signal temporal logic (STL) formulas [42] and correct syntax errors if detected. It then uses an STL planner to generate trajectories. By using an intermediate formal task specification, AutoTAMP outperforms LLM planners on tasks with geometric and temporal constraints in 2D domains.

Recent work also leveraged LLMs for grounding natural language commands to first-order logic (FOL) formulas. LEFT [19] used an LLM to translate natural language queries to FOL programs, which a differentiable FOL executor executed. At the same time, a domain-specific grounding model grounded the symbols of the FOL program to various input modalities, e.g., 2D images and point clouds. The advantage of using first-order logic over LTL is capturing commands that use quantifiers and predicates for generalization. On the other hand, LTL provides a natural way to concisely represent temporally extended commands. Different logic representations can be formally composed to form a new, more expressive logic representation. These representations can enable direct mapping between

abstract concepts in a language such as "avoid" and a precise, formal logical expression that guarantees task satisfaction.

2.2.2 Planning Domain Definition Language (PDDL)

The Planning Domain Definition Language (PDDL) [43, 44, 45, 46] is a structured representation that defines a planning problem. It consists of a domain, which defines objects, predicates, and actions that govern the world's rules, and a problem, a grounded problem instance with an initial state and a desired goal state. A symbolic planner takes as input the PDDL domain and problem and then outputs a plan [47], i.e., a sequence of actions, to reach the goal state from the initial state. In this sense, it is a goal-based representation, but because it encodes actions and their effects, it can also be used imperatively. Recent works used LLMs to translate natural language descriptions of the world and the task to a PDDL representation of the planning problem, which a symbolic planner can then use. Compared to logical representations, PDDL provides a language of predicate states and transitions that allow goals to be translated into lower-level actions and skills but requires a full domain specification. People have defined PDDL domains for a wide variety of real-world applications. [48] showed how a robot can learn symbols for low-level skills that are both necessary and sufficient to enable planning in PDDL.

[21] prompted LLMs with a PDDL domain description, an initial state, and examples to translate a natural language goal description to a PDDL goal specification. Results in simulation showed that LLMs could translate unambiguous goal descriptions and fill in missing details for under-specified goals but struggled with numerical and spatial reasoning. [49] also showed that translating natural language goal descriptions to PDDL goals and then solving them using a symbolic planner outperformed directly using an LLM as a planner.

[50] and [22] prompted an LLM to translate a natural language description of a problem into a complete PDDL problem definition, which was then fed into a symbolic planner

together with a PDDL domain definition to find an optimal plan. These approaches outperform LLM planners and provide strong correctness guarantees from using a symbolic planner. Recent work has also created benchmarks for evaluating LLM's ability to make plans concerning PDDL and other AI baselines [51, 52], finding that LLMs achieve low success rates across domains in isolation. Still, they can improve the search process for underlying sound planners, leading to higher performance.

Unlike other works that use LLMs for AI planning, [53] developed a generalized planner to synthesize Python programs to solve novel tasks by prompting LLMs with a domain specification and a few training tasks emphasizing satisfaction and efficiency. Their system also automatically detects planning errors and then re-prompts the LLM with feedback.

2.2.3 Code

Code is a very flexible form of formal representation that can be used in a goal-based or an action-based manner. Indeed, a Python program can be generated that directly outputs motor torques for robots or implements an arbitrary deep-learned function. A formal language is typically specified as a subset of the language at hand using a manually defined programming API consisting of functions and their arguments analogous to robot skills. Skills or functions are not simply linearly called but can be embedded in more complex logic, like conditionals and loops.

Code as Policies [8] prompted an LLM with import statements, example code, and code comments that describe the desired policy to generate Python code executable directly on the robot. It solved manipulation and mobile manipulation tasks with API calls to Python libraries and predefined perception and control modules. ProgPrompt [23] applied a similar approach with additional assertion statements to recover from errors when reliable state tracking is available. [54] developed a modular bi-arm system that employs an LLM to map natural language instructions to high-level API calls to manipulation skills powered by VLMs and a control module to solve three tabletop bimanual manipulation tasks. Their

experiment results highlighted the benefits of modularity for ensuring safety, interpreting failures, and identifying modules to improve. Socratic Models [55] also demonstrated the code generation capabilities of LLMs for simulated pick-and-place tasks, provided with pre-trained perception modules and robot policies. ITP [25] used LLMs to generate high-level action sequences and then translated the action descriptions to predefined function calls using APIs for an object detector and robot policies to solve tabletop manipulation tasks. By storing completed high-level actions, ITP can replan given new commands at any step during the execution. Voyager [24] extended the code generation capabilities of LLMs to build a lifelong learning agent in Minecraft that continuously explores and learns a skill library of executable code.

2.2.4 Predefined Skills

Recent methods have used LLMs as a planner to map language commands to a sequence of predefined skills. Skills can be learned from data or manually specified. These predefined skills are a formal representation since they are discrete and manually specified. (Even if skills are learned, they are frequently learned from human-provided demonstrations, which provide the discrete structure for the skills.) They are also action-based representations specifying actions to take rather than goals or resultant states. For example, a robot might be provided with skills such as "pick up," "put down," "drive to refrigerator," "drive to microwave," and "clean table." Then, the challenge is to map a natural language instruction such as "Get me the apple" and "Clean the table with the sponge" to a sequence of skills. [56] iteratively prompted an LLM to decompose a high-level task specification in natural language to a sequence of action descriptions. They used sentence similarities to map the proposed action descriptions to actions available in a simulated environment where their corresponding predefined low-level controllers can be executed. SayCan [9] iteratively prompted an LLM to sequence pre-trained skills described by predefined verb phrases based on their probabilities of success from the current state to solve mobile

manipulation tasks specified by natural language on a physical robot. Their approach relies on both formal representations and high-dimensional, end-to-end differentiable representations. Despite using a fixed ontology of skills, these are implemented using a deep approach. SayCan learns these skills using a multi-task value function conditioned on language. CAPE [57] also used an LLM planner to sequence predefined skills. When a plan does not meet the precondition of some skill, it re-prompted the LLM with corrective feedback. In addition to using an LLM to ground language to predefined skills, Inner Monologue [58] also enabled language feedback based on pre-trained perception models for describing the scene and detecting successful execution of skills. In these methods, the specific skills are often given relatively little attention, yet having the right set of skills at the right level of granularity is critical to success. We categorize this method on the formal side of the spectrum because the skills impose significant structure. That said, it imposes less structure than goal-based methods such as logical representations, leaving the LLM to keep track of higher-level constructs such as constraints, sequencing, and conditionals.

2.3 Mapping Language to High-Dimensional Vectors to Actions

On the other end of the spectrum are approaches that map natural language instructions to high-dimensional, non-formal representations. These deep approaches are largely defined by data and learning, whereas formal, symbolic representations are largely human-crafted and invoke manually provided structure. The large models that enable deep and many symbolic approaches are created by self-supervised pretraining on large datasets scraped from the web. These models thereby learn a generative representation of images, text, and other modalities that capture background and task knowledge that is useful for robotic applications, for example, as in [29]. By learning to model a large and varied distribution,

the models also learn to perform tasks useful for robotic language grounding, including translation, semantic parsing, and broadly useful ones like in-context learning [59].

Pretrained models can represent high-level and low-level semantics, so they can represent high-level instructions and low-level actions. This semantic understanding can extend across modalities while retaining these task-learning abilities, forming combined representations of language and vision useful to robotics problems. Beyond their representation capacity, a unified interface for training and inference makes these methods particularly powerful. Multiple models across modalities can be connected and trained end-to-end through gradient descent, and pretrained representations can be improved through increases in model and data scale [59, 60, 61, 62]. Provided with a method for generating and standardizing data, a single pipeline with minimal human intervention can be used for continued improvement and deployment. One limitation of deep methods is that they require abundant data, which limits their applicability to domains where data is easy to collect. However, large-scale pretraining can significantly reduce the amount of task-specific training data required [59].

In robotic language grounding, deep learning approaches map input language commands to one of the following representation types: actions represented as low-level controls (policies) or high-level goals represented as natural language, images, or neural network activations. These high-level goals are used to condition low-level actions from planners or policies to drive robot actions. At the lowest level, the approaches map language to sequences of joint states, end-effector poses, or motor torques. High levels of abstraction are easier for humans to interpret versus low-level joint states, making it difficult to discern the outcome of the action. We order from high-level to low-level to indicate the levels of abstraction as they have evolved in the field and to create a spectrum for interpretability.

2.3.1 Image and Language Subgoals

These methods use images and natural language to express subgoals. [27] perform pick-and-place tasks on a robot by alternating between generating images representing subgoals using a text-guided image-editing model and executing a low-level policy conditioned on the goal images. They map language to subgoals represented as images and then map these images to end-effector positions. VLP [28] transforms the input image and command into a sequence of language and image subgoals and then generates plan rollouts with tree search using a language-conditioned video generation model. It was deployed on a robot to perform tabletop arrangement tasks.

UniSim [26] trains a generative video model to predict the outcome of both high and low-level actions. Using this model, the authors train high-level planners and low-level policies in simulation and demonstrate zero-shot transfer to real-world autonomous driving tasks. Language and images are mapped to high-level skills and low-level controls, e.g., "move the gripper to (x, y)". GAIA-1 [63] similarly leverages real-world driving data to develop a world model that generates life-like driving behavior by predicting outcomes in the video space. Coarse symbols, such as words in English, are not as rich of a representation as images. Incorporating multimodality increases the generalizability and expressiveness of the systems' states. This allows for directly representing the visual world state without introducing an intermediate representation like natural language, scene graphs, or maps. The downside to using images is that it requires more pretraining and fine-tuning data to learn this multimodal representation.

2.3.2 End-Effector and Joint-State Goals

These approaches use low-level skills and parameterize actions using joint states or end-effector position, in contrast to high-level skills like "pick up" that we classify on the formal side of the spectrum. Often, they are not learned because they are sufficiently low-level, and the built-in controllers on the robot are sufficient. For example, the RT-*

family of models [31, 32, 33] use an action space that specifies the arm, base, and end-of-episode token. For the arm, they use the end-effector pose of the gripper (vs. lower-level joint states or joint torques) and rely on inverse kinematics to find joint states to move to the end-effector pose.

RT-1, RT-2, and RT-X [31, 32, 33] perform various language-conditioned tasks on 22 distinct robotic platforms. They collected a large and diverse dataset of demonstrations and trained a large multimodal transformer that maps language and state observations to low-level discrete end-effector controls. These methods attempt to demonstrate positive transfer, the transfer of knowledge between tasks that allows a system to perform better at both tasks simultaneously. The Open-X Embodiment dataset [33] release came with baselines showing this phenomenon, whereas GATO [64] did not. Open-X Embodiment targeted end-effector positions and used models from 35M to 55B parameters, whereas GATO targeted joint states directly and used 1.2B parameters. Octo [65] also shows better performance across tasks by leveraging the Open-X Embodiment dataset while simultaneously outputting end-effector positions and joint-state control. PaLM-E [29] is an image and text multimodal model finetuned end-to-end for various multimodal reasoning tasks, including robot manipulation. This model maps visual-language input describing a scene to a limited vocabulary of actions, which are then turned into robot actions. VIMA [10] learns a multimodal model that maps tasks specified with images and language to high-level actions parameterized by low-level end-effector controls to perform tabletop arrangement tasks on a simulated robot. One consequence of this level of abstraction is that what is learned is often specific to a robotic platform and cannot be generalized from one environment and robot to another without more data. Generating data automatically in simulation allowed it to overcome limitations on the data required to learn a meaningful representation. Future work is focused on increasing the size and diversity of the datasets to enable more generalization.

Recent works in foundation models have shown tremendous improvement in gener-

alization by leveraging large amounts of data. These methods often use symbols in the form of vector-quantized image patches to reduce the number of tokens required [66]. ALOHA [67] seeks to address the limited data for training these systems by utilizing accessible hardware for many research institutions. They then use an end-to-end foundation model that targets joint-state control of robots to execute policies on the bimanual robot platform. [68] addresses domain limitations by leveraging vision and text models that are larger and trained on more data. [30] develop a language-conditioned behavior-cloning model that encodes the inputs of a natural language command and a voxel grid to predict a collision-free pose that a motion planner can execute. Similarly, VPT [69] utilizes real demonstrations of users playing Minecraft to train better policies that target a discrete set of actions for the agent to follow. They expand their dataset by training an inverse dynamics model to label data from YouTube. Joint-state and end-effector goals are useful for simultaneously targeting multiple robotic platforms but can lack the interpretability of their output.

2.4 Discussion and Future Directions

Each end of the spectrum has tradeoffs and advantages, and the two approaches are complementary in many ways. As shown, many recent works combine aspects of symbolic and end-to-end methods. A number of open problems and exciting future research directions are to leverage the best of both worlds.

2.4.1 Representation

Table 2.1 summarizes the grounding representations used by various systems we review and how they execute their output on a robot in various simulated and real-world domains. On the formal side of the spectrum, system output ranges from more structured logic formulas and code to less structured predefined skills. On the deep side of the spectrum, methods ground language input to high-dimensional representations of joint states or

end-effector poses. Some deep approaches first ground language to an intermediate representation of image or language subgoals, then low-level controls. Formal approaches to executing the system output on a robot use a planner or a code interpreter together with low-level controllers. In contrast, deep approaches use low-level controllers to produce desired robot motion. A key takeaway is using intermediate representations at different system levels, which enables the use of "off-the-self" robot modules such as SLAM and object detectors. When using these modules, there is a crisp abstraction barrier between the learning model and the robot, limiting what needs to be learned and the flexibility of the learned system. Automatically learning an extendable set of symbols and grounding those symbols [70] can address this limitation of formal approaches. Another key research question for formal representations is what formal language to use. Yet a formal language that is powerful and flexible enough to capture all English or other natural languages is unknown. Approaches such as RLANG [71] expand the scope of what language can talk about (from goals and actions to observations, states, and transition functions). Integrating these formal, manually specified languages with open-class learned models is critical, for example, jointly learning skills and symbols [48, 70]. This survey only focuses on natural language, yet using multimodal input, e.g., text, audio, RGB-D images, videos, and joint trajectories, etc., to complement language [64] in solving robotic tasks is also a promising future research direction. Current state-of-the-art approaches combine multiple modalities in a single neural architecture capable of running in real-time, including speech, image, and text [72]. This promises to enable more audio interactivity between humans and robots.

2.4.2 Situated Natural Language Commands

Table 2.2 reviews various domains and natural language commands used to evaluate each system we review. Domains vary from tabletop manipulation tasks, usually focusing on pick-and-place, to mobile manipulation, usually involving chaining pick-and-place

actions and navigation, in both simulation and the physical world. Table 2.2 shows examples of natural language commands that specify temporally extended tasks and spatial relations among objects in various domains. Natural language was created based on an assumption about human reaction time. The methods described in this work have control loops in the space of 1-3 Hz, which cannot close the loop fast enough to react to sentences such as "move to the left" followed by "okay, stop." For real-time robotics applications, faster language processing methods need to be created. These could be through the advent of faster neural network hardware or methods that learn hierarchical abstraction at different frame rates to support the cross-cutting ability of language to talk about any part of the robot system.

Command Types	Domains	Command Examples	Implementation Examples
Temporal	Navigation	"Move back and forth between the table and the countertop twice."	[8]
		"Go to the store on Main Street, but only after visiting the bank"	[20]
		"Remain on the first floor and navigate to the red room ."	[34]
	Manipulation	"Scan for blocks and insert any found into the bin."	[73]
		"Create a stack that contains two blocks."	[21]
	Mobile Manipulation	"Get Glass of Milk"	[56]
		"Always take the pear and go to the tree and stay there."	[35]
		"I spilled my coke. Can you bring me something to clean it up?"	[9]
		"Can you bring me the drink from the table?"	[58]
		"Bring the plate to the kitchen"	[37]
		"Move KeyChain1 to the box with more books"	[21]
Spatiotemporal	Navigation	"Move to the white car near the apartment at most twice, in addition, avoid the dumpster near the white car" "Move back and forth between Object A in Room B and Object C in Room D"	[20, 34]
	Manipulation	"Put the red block to the left of the rightmost bowl"	[8]
		"Pack the ring into the brown box"	[19]
		"Sort fruits on the plate and bottles in the box"	[23]
		"Move all the blocks to different corners."	[55]
		"May I have a cup of milk with taro?"	[25]
		"Stack Objects X, Y, and Z" "Push Object X to Object Y"	[8, 10, 26, 27, 27, 28, 29, 31, 32, 33]
	Mobile Manipulation	"Take the Coca-Cola can from the desk and put it in the middle of fruits on the table"	[8]
		"Microwave salmon"	[23]
		"Bring Object A to Location B;" "Open the door and proceed to the kitchen"	[9, 29, 37, 58]
		"Build a house out of dirt blocks"	[24]
		"Put away groceries in the pantry"	[56, 74]

Table 2.2: Situated Natural Language Commands

2.4.3 Datasets

Both approaches require datasets for learning and evaluation. Formal methods generally require parallel datasets that map natural language to structures in the formal representation or at least the ability to test if a formal representation is correct. These parallel corpora can be expensive at scale, but new LLM methods enable good performance even with much smaller datasets. A common approach is to show a trajectory of robot behavior generated from an underlying formal representation and then ask annotators to describe

the trajectory in language [36, 73]. This approach can enable untrained annotators to provide parallel data. Still, it often leads to ambiguous situations because the trajectory does not overtly show the constraints present in the underlying representation. Thus, it remains an open problem to collect diverse and unambiguous language commands and their corresponding labels.

To compensate for this problem, approaches, especially less formal approaches, try to learn from large unannotated datasets or from datasets obtained from simulation. LLMs can also enable robust learning to map from natural language to formal representation using few-shot learning or fine-tuning [8, 9, 19, 20, 22]. Methods that use less constrained representations, such as VIMA [10], collect data in simulation across many scenarios and eschew a formal representation that requires annotation of any kind, instead directly train from joint trajectories, images, and video.

2.4.4 Generalization and Bias

Models that map language to a structured representation can exploit the regularities of that structure to train from smaller datasets and can generalize by porting the formal representation to different domains. Depending on the domain, this generalization can be significant by relying on state-of-the-art robot algorithms such as SLAM and motion planning rather than trying to learn everything end-to-end. This ability is possible because of the strict, modular compositionality introduced by the formal model.

On the other hand, end-to-end models have the potential to be applied in domains given large enough training datasets and computational resources. Looking at the power of LLMs, one predicts that a multimodal dataset of video, other robot sensor data, and joint states may be able to generalize as flexibly and powerfully as an LLM does with language. However, training this model requires significantly higher dimensions than language data and potentially much larger dataset sizes. However, if provided with general-purpose robot data, foundation models have the potential to understand language

in a very general way, analogous to the success of LLMs. Key research challenges to address this problem include acquiring semantically diverse datasets covering desired robot behaviors, developing sample-efficient model architectures and training algorithms for multimodal datasets many times larger than those used to train LLMs, and translating learned models to robot action.

Many existing works that use foundation models also leverage symbols, but do not speak to the symbols' role in building practical systems. We observe that in many recent RoboNLP papers, including [9, 10, 31, 32, 33], all leverage symbolic representations, either in the form of discretized action spaces or predefined skills. We would like to see more authors acknowledge that their work is made better because of the symbolic representation their model uses or propose how other symbolic representations can improve the model.

In the future, we expect models to better represent the world through these finer-grained symbolic representations. After a certain point, combining text, audio, color images, depth images, point clouds, etc., becomes functionally continuous. While the POMDP and PDDL methods are theoretically sound, in practice, they lack representational capacity outside of their domain due to the brittleness of out-of-domain representation. We believe that future successful methods will leverage deep learning methods with smarter representations of the input and output data via transformations of the intermediate representations. This includes representing color images as point clouds, maps, or meshes as derivative intermediates. We can then augment these with state representations of the environment as text or other labels. There is an eventuality where the bitter lesson [75] takes over, and throwing even more parameters and data at the problem will win out, but that is predicated on an evergrowing data supply. It remains to be seen where additional data will come from and if methods that leverage only partial observations from one modality can synergize multiple modalities simultaneously.

2.4.5 Limitations of Natural Langauge

As robot operators, we want to be able to express the state of our robotic system in a way that is auditable by us and useful to the robot in accomplishing tasks. Many of the methods we discussed also focus on English as the representation of natural language encoded in LLMs and multimodal models. While English works well for much of the world, it does not precisely describe objects' physical locality. Saying that an apple is to the left of an orange can put that apple in various positions with just a language description alone. While adding geometric priors could enhance the specificity of the language, we should, as a field, look to building benchmarks to evaluate the physical plausibility of these language models as they operate in different languages [76, 77]. Robot morphology and human morphology are very different. It is likely that expressing tasks as being completed by one's arm, such as "clean the dishes with your left hand" may be completely moot to a robot that uses soft-body mechanics to interact with the world. Human language evolved in a biological domain that necessitates a more survival vocabulary. These actions are not necessarily helpful for describing robot actions and could contribute to skewing the success rates of robots in accomplishing tasks.

2.4.6 Safety and Interpretability

Interpretable and explainable robotic systems provide transparency in decision-making and gain trust in human-robot interaction [78, 79, 80]. Moreover, verifiable safe operation is essential for deployments that satisfy worldwide standards such as ISO 61508 [81], which defines standards for safely deploying robots in industrial factory environments worldwide. These standards require that robotic systems be mathematically proven to have a failure rate lower than 10^{-5} dangerous failures per hour. Robots should also provide feedback to unsatisfiable task specifications [82] or explain their actions when execution fails [83]. Safety can be classified into semantic and kinematic/physical safety [54]. Examples of semantic safety are never entering the nursery and always transporting a cup full of

coffee in an upright orientation. Examples of kinematic safety are avoiding collisions with humans and objects and avoiding reaching joint/velocity/torque limits. Formal approaches using LLMs mostly focus on enforcing semantic safety. Many works on formal methods and safe control do not use LLMs and encode kinematic safety in trajectory optimization [84]. For example, linear temporal logic (LTL) has been used extensively to develop provably safe controllers [85, 86]. Given a logical specification and the system model, formal methods can synthesize correct-by-construction robot controllers or provide counterexamples to explain when the task is unsatisfiable with respect to how well the model captures reality [12]. A key challenge for any safety framework of this kind is grounding predicates in real-world noisy and partial perceptual data.

Existing deep learning approaches do not have a clear pathway for achieving this level of interpretability and safety. If we look at human behavior, defining a set of safety guidelines to assess adherence has been challenging. While we can continue to apply formal representations to these models to varying levels of success, there remains a need for a system that, at a meta-level, operates on symbols but whose abilities are as general and flexible as humanity's. For example, one of our recent papers uses a combined approach with LLMs and a formal representation, showing that we can exploit the generalization ability of LLMs with the formal safety guarantees provided by LTL to create a safe yet robust and flexible system for following commands [87]. Yet this approach only scratches the surface. Much more needs to be done to integrate these systems, especially to study the interpretability and safety guarantees inherent in perceptual systems and provide formal guarantees and bounds about the behavior of deep networks.

2.5 Conclusion

Our review characterizes the literature in robotic language grounding along a spectrum from using more formal, biased, discrete representations to less formal, less biased, higher-dimensional continuous representations. There are benefits and tradeoffs to each approach.

More formal methods induce structure that can limit the size of the learning problem and provide interpretability and formal safety guarantees. However, they also constrain the output space, limiting the flexibility and expressive power of what can be learned. Less formal methods impose fewer constraints but require more data and possibly more structured neural networks to be learned. Methods such as SayCan [9], traditionally considered less structured, use a formal representation of the robot's skills. Seen in this context, it is clear that a limitation for all methods is the lack of physical capability of existing robots: a key area of future work is to enable robots to perform a larger variety of tasks in a larger variety of environments. Hence, they can physically perform the tasks people ask them to do.

CHAPTER 3

Grounding Temporal Language

In Chapter 2, we propose some desired properties of a language grounding representation, including having unambiguous semantics, being compositional, being able to represent temporally-extended tasks, and provide formal safety guarantees. In this chapter, we show that 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 task 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. Code, datasets, videos, and supplementary materials are at `lang2ltl.github.io`.

Figure 3.1: Lang2LTL can ground complex navigational commands in household and city-scaled environments without retraining.

3.1 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 [88] 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 [35, 89, 90, 91]. In contrast, some recent approaches proposed leveraging LLMs to directly generate robot policies [8, 92, 93, 94]. The LLM output is itself a formal language that must be interpreted by an external 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 [95, 96, 97, 98, 99]. 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 2,125 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.

3.2 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 [100] pretrained to predict the next token given a context window [101, 102]. In this work, we used the GPT series models [103, 104] and the T5-Base model [105].

Temporal Task Specification: Linear temporal logic (LTL) [88] has been the formalism of choice for expressing temporal tasks for a variety of applications, including planning and reinforcement learning [95, 96, 99, 106, 107, 108], specification elicitation [89, 90, 109, 110, 111], and assessment of robot actions [112]. The grammar of LTL extends propositional logic with a set of temporal operators defined as follows:

$$\varphi := \alpha \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \mathbf{X}\varphi \mid \varphi_1 \ \mathbf{U} \ \varphi_2 \tag{3.1}$$

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 [113]. The Additional Details Section 3.9.1 lists the specification patterns and their interpretations.

Planning with Temporal Task Specification: Every LTL formula can be represented as a Büchi automaton [114, 115] 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 [95, 96, 97, 98, 106, 107]. Lang2LTL is compatible with any planning or reinforcement learning algorithm that accepts an LTL formula as task specification.

3.3 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 [116, 117, 118, 119, 120]. Recent work leverages large pretrained models to directly generate task plans, either as code [8, 92] or through text [94, 121]. 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 [93] 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. [113], while Lang2LTL can interpret 15 temporal tasks.

Translating Language to Formal Specification: Prior approaches are tied to the domain they were trained on and require retraining with a large amount of data to deploy in a novel domain. Gopalan et al. [90] relied on commands paired with LTL formulas in a synthetic domain called *CleanUp World*. Patel et al. [91] and Wang et al. [35] 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 [34, 40, 122, 123]. 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. [89], uses CopyNet [124] architecture to generate LTL formula structures followed by explicitly replacing the appropriate propositions. We demonstrate a significant performance improvement over Berg et al. [89]'s approach by leveraging LLMs as well as training and evaluating on a semantically diverse dataset.

3.4 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. The Additional Details Section 3.9.2 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 [97] for this paper.

Consider the example of a drone following a given command within an urban environment depicted in Figure 3.2. 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.

3.5 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

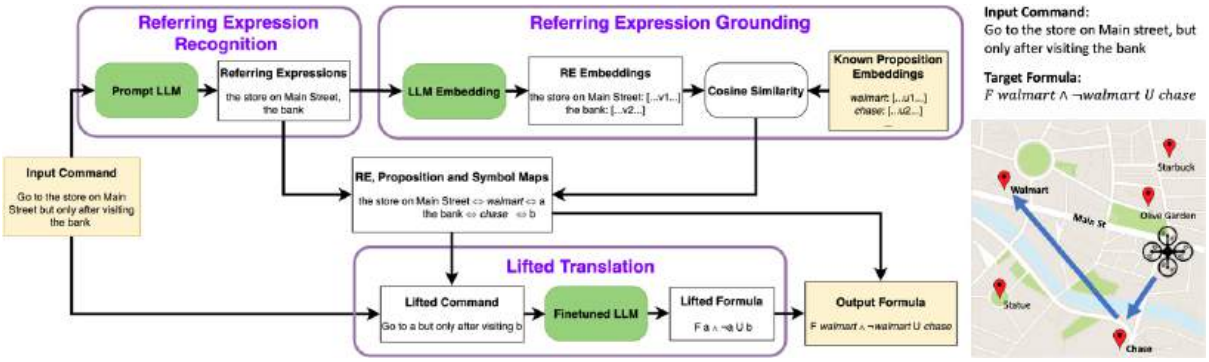


Figure 3.2: Lang2LTL system overview: Green blocks are pretrained or finetuned LLM models. Yellow blocks are the input or output of the system.

the command u that refer to Boolean propositions. In this case, $\{r_i\} = \{$" the store on Main Street", "the bank"$\}$.

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 3.2.

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 3.6.

3.5.1 Referring Expression Recognition (RER)

Referring expressions are noun phrases, pronouns, and proper names that refer to some individual objects [125]. 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 the Additional Details Section 3.9.4.

3.5.2 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 [126].

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. [127], 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,

$$k^* = \underset{\{k:(z,f)\}\in\mathcal{D}}{\arg\max} \frac{g_{embed}(r_i)^\top g_{embed}(z)}{||g_{embed}(r_i)||\ ||g_{embed}(z)||} \tag{3.2}$$

3.5.3 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. [90] and Patel et al. [91] 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 to shorten the output.

The lifted translation module accepts a lifted utterance as an input and generates a lifted 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 the Additional Details Section 3.9.5.

Finetuned LLM: We finetuned LLMs using supervised learning following [101]. We tested two LLMs with supervised finetuning, namely, T5-Base (220M) [105] (using the Hugging Face Transformer library [128], and the text-davinvi-003 version of GPT-3 using the OpenAI API. The target was an exact token-wise match of the ground-truth LTL formula.

Prompt LLM: We evaluated prompting the pre-trained GPT-3 [103] and GPT-4 [104] 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 trans-

former architecture [100] 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.

3.6 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 3.6.1. We evaluated the performance of each module described in Section 3.5 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 [129].

3.6.1 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, $u_1 =$" Go to *chase*", and $u_2 =$"Visit *chase*," describing the same temporal formula **F** *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 generalization 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 [34, 35, 40, 89, 90, 91] demonstrate robustness to paraphrasing.

Robustness to Substitutions: Assume the system has been trained to interpret the command corresponding to **F** *chase*, and **G** $\neg$*walmart* at training time but has not been trained on a command corresponding to **F***walmart*. If the system correctly interprets the command corresponding to **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. [90] demonstrated limited robustness to substitutions. The lifted translation approach followed by Berg et al. [89], NL2TL [40], and Lang2LTL demonstrates robustness to substitutions

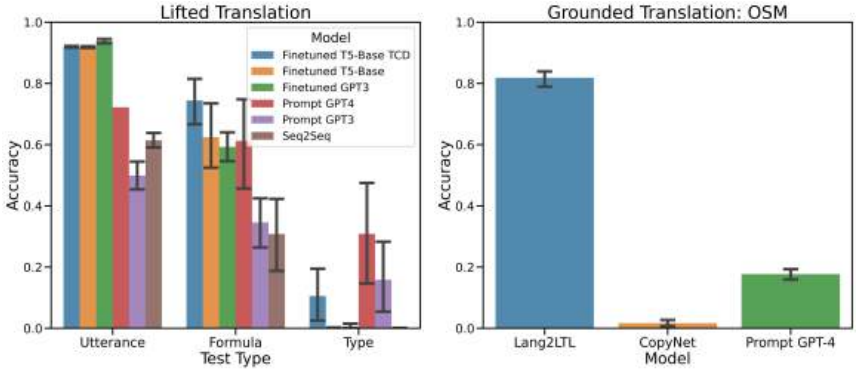
Robustness to Vocabulary Shift: Assume the system has been trained to interpret commands corresponding to **F***chase* at training time but has never seen any command containing the propositions *walmart*. If the system correctly interprets **F***walmart* at test time, the system is said to be robust 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 [40, 93, 127]. Our full system evaluations demonstrated robustness to novel vocabularies in Section 3.6.5.

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 **F** *chase*, and **F** *walmart* are transformed to **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 built-in equivalency checker from the Spot LTL library [130].

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 [113, 131], 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 the 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 3.3: Figure 3.3a depicts the average accuracies of six lifted translation models on the three holdout sets over five-fold cross-validation. Figure 3.3b depicts the average accuracies of the grounded translation task in the *OSM* domain.

Table 3.1: Proposition Grounding Evaluation

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

Table 3.2: Cross-Domain Evaluation. * for Zero-Shot

	OSM [127]	*CleanUp* [90]
Lang2LTL	$49.40 \pm 15.49\%^*$	$78.28 \pm 1.73\%^*$
CopyNet [127]	$45.91 \pm 12.70\%$	$2.57\%^*$
RNN-Attn [90]	NA*	$95.51 \pm 0.11\%$

3.6.2 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. [113] 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, please refer to the Additional Details 3.9.8.

3.6.3 Grounded Dataset

Generating Diverse REs: We used GPT-4 to paraphrase landmark names from an open-source map database, *OpenStreetMap* (*OSM*) [129], to more diverse forms of referring expressions (REs) by providing the semantic information. The prompt used for generating diverse REs is in the Additional Details 3.9.3.

We then substituted the symbols in the lifted dataset (Section 3.6.2) by diverse REs on 100 randomly sampled lifted utterances for each of the 21 *OSM* cities. For a list of example commands, please see the Additional Details Table 3.7.

3.6.4 Component-wise Evaluation

Proposition Grounding: We evaluated the RER and REG modules on the grounded *OSM* dataset with 2,100 utterances across 21 cities. The average accuracy and the standard error across the cities for these modules are depicted in Table 3.1. The accuracy of RER decreased slightly, and REG performed uniformly well as we varied the complexity of commands and REs, respectively (the Additional Details Figure 3.9).

Lifted Translation: We evaluated the six models presented in Section 3.5.3 for lifted translation through five-fold cross-validation on *utterance*, *formula*, and *type holdout* tests. Figure 3.3a 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

type-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.

3.6.5 System Evaluation

We compared the translation accuracy of the full Lang2LTL system on the grounded datasets with the CopyNet-based translation model [89] and Prompt GPT-4. We retrained CopyNet with an identical data budget as the 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 3.3b 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.

3.6.6 Cross-Domain Evaluation

We further tested the zero-shot generalization capability of Lang2LTL on two different crowd-sourced datasets from prior work; the Cleanup World [90] on an analog indoor environment; and the *OSM* dataset [89] collected via Amazon Mechanical Turk. Table 3.2 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. [89]. 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 needs to generalize to unseen formulas.

3.7 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

[132] with the AP-MDP planner [97] 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 [97] planner can directly plan over this semantic map, given an LTL task specification. Please refer to the Additional Details 3.9.10 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) [8], 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 ground 52 commands (40 satisfiable and 12 unsatisfiable commands). The failure cases were due to incorrect lifted translations. 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. CaP demonstrated more robustness to paraphrasing than our system, which failed on some compositional patterns not in the augmented lifted dataset.

3.8 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 retraining 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.

3.9 Additional Details

3.9.1 Specification Patterns

We developed Lang2LTL to ground navigational commands to LTL formulas. We started with the catalog of robotic mission-relevant LTL patterns for robotic missions by Menghi et al. [113]. We adopted 15 templates that are relevant to robot navigation and modified some of their patterns to semantically match our requirements. The complete list of pattern descriptions and the corresponding LTL templates is in Table 3.3. Note that we use some additional abbreviated temporal operators, specifically "Weak until" $\mathbf{W}$, and the "Strong release" $\mathbf{M}$ in terms of standard operators, i.e., $a \ \mathbf{W} \ b = a \ \mathbf{U} \ (b \ \vee \ \mathbf{G}a)$, and $a \ \mathbf{M} \ b = b \ \mathbf{U} \ (a \ \wedge \ b)$.

3.9.2 Semantic Information from OpenStreetMap Database

An example entry of the semantic dataset is as follows,

```
{
    "Jiaho supermarket": {
        "addr:housenumber": "692",
```

Table 3.3: Specification Patterns for Lang2LTL

Specification Type	Explanation	Formula
Visit	Visit a set of waypoints $\{p_1, p_2 \ldots, p_n\}$ in any order	$\bigwedge_{i=1}^{n} \mathbf{F}\ p_i$
Sequence Visit	Visit a set of waypoints $\{p_1, p_2 \ldots, p_n\}$, but ensure that p_2 is visited at least once after visiting p_1, and so on	$\mathbf{F}(p_1 \wedge \mathbf{F}(p_2 \wedge \ldots \wedge \mathbf{F}(p_n)) \ldots)$
Ordered Visit	Visit a set of waypoints $\{p_1, p_2 \ldots, p_n\}$, but ensure that p_2 is never visited before visiting p_1	$\mathbf{F}(p_n) \wedge \bigwedge_{i=1}^{n-1}(\neg p_{i+1}\ \mathbf{U}\ p_i)$
Strictly Ordered Visit	Visit a set of waypoints $\{p_1, p_2 \ldots, p_n\}$, but ensure that p_2 is never visited before visiting p_1, additionally, ensure that p_1 is only visited on a single distinct visit before completing the rest of the task	$\mathbf{F}(p_n) \wedge \bigwedge_{i=1}^{n-1}(\neg p_{i+1}\ \mathbf{U}\ p_i) \wedge \bigwedge_{i=1}^{n-1}(\neg p_i\ \mathbf{U}\ (p_i\ \mathbf{U}\ (\neg p_i\ \mathbf{U}\ p_{i+1})))$
Patrolling	Visit a set of wwaypoints $\{p_1, p_2 \ldots, p_n\}$ infinitely often	$\bigwedge_{i=1}^{n} \mathbf{GF} p_i$
Bound Delay	If and only if the proposition a is ever observed, then the proposition b must hold at the very next time step	$\mathbf{G}(a \leftrightarrow \mathbf{X}b)$
Delayed Reaction	If the proposition a is ever observed, then its response is to ensure that the proposition b holds at some point in the future	$\mathbf{G}(a \rightarrow \mathbf{F}b)$
Prompt Reaction	If the proposition a is ever observed, then the proposition b must hold at the very next time step	$\mathbf{G}(a \rightarrow \mathbf{X}b)$
Wait	The proposition a must hold till the proposition b becomes true, and b may never hold	$a\ \mathbf{W}\ b$
Past Avoidance	The proposition a must not become true until the proposition b holds first. b may never hold	$\neg a\ \mathbf{W}\ b$
Future Avoidance	Once the proposition a is observed to be true, the proposition b must never be allowed to become true from that point onwards.	$\mathbf{G}(a \rightarrow \mathbf{XG}\neg b)$
Global Avoidance	The set of propositions $\{p_1, p_2 \ldots, p_n\}$ must never be allowed to become true	$\bigwedge_{i=1}^{n} \mathbf{G}(\neg p_i)$
Upper Restricted Avoidance	The waypoint a can be visited on at most n separate visits	For $n = 1$, $\neg\mathbf{F}(a \wedge (a\ \mathbf{U}\ (\neg a \wedge (\neg a\ U\ \mathbf{F}a))))$ For $n = 2$, $\neg\mathbf{F}(a \wedge (a\ U\ (a \wedge (\neg a\ \mathbf{U}\ \mathbf{F}(a \wedge (a\ \mathbf{U}\ (\neg a \wedge (\neg a\ \mathbf{U}\ Fa))))))))$
Lower Restricted Avoidance	The waypoint a must be visited on at least n separate visits	For $n = 1$, $\neg\mathbf{F}a$ for $n = 2$, $\mathbf{F}(a \wedge (a\ \mathbf{U}\ (\neg a \wedge (\neg a\ U\ \mathbf{F}a))))$
Exact Restricted Avoidance	The waypoint a must be visited on exactly n separate visits	For $n = 1$, $a\ \mathbf{M}\ (\neg a \vee \mathbf{G}(a \vee \mathbf{G}\neg a))$ For $n = 2$, $(a \wedge \mathbf{F}(\neg a \wedge \mathbf{F}a))\mathbf{M}(\neg a \vee \mathbf{G}(a \vee \mathbf{G}(\neg a \vee \mathbf{G}(a \vee \mathbf{G}\neg a))))$


```
        "shop": "supermarket",
        "opening_hours": "Mo-Su 08:00-20:00",
        "phone": "6173389788",
        "addr:postcode": "02111",
```

```
        "addr:street": "Washington Street"
    },
    ...,
}
```

3.9.3 Implementation Details about Referring Expression Generation

We prompt the GPT-4 model for paraphrasing landmarks with corresponding OSM databases. For each landmark, three referring expressions are generated. The prompt for generating referring expressions by paraphrasing is as follows,

Referring Expression Generation Prompt

```
Construct a short natural language phrase to describe the landmark provided in a
Python dictionary.

Landmark dictionary:
'Fortuna Cafe': {'addr:housenumber': '711', 'cuisine': 'chinese', 'amenity':
'restaurant', 'addr:city': 'Seattle', 'addr:postcode': '98104', 'source': 'King
County GIS;data.seattle.gov', 'addr:street': 'South King Street'}
Natural language:
Chinese cafe on South King Street

Landmark dictionary:
'Seoul Tofu House & Korean BBQ': {'addr:housenumber': '516', 'cuisine': 'korean',
'amenity': 'restaurant', 'addr:city': 'Seattle', 'addr:postcode': '98104',
'source': 'King County GIS;data.seattle.gov', 'addr:street': '6th Avenue South'}
Natural language:
Seoul Tofu House

Landmark dictionary:
```

```
‘AI Video’: {‘shop’: ‘electronics’}
Natural language:
AI Video selling electronics

...

Landmark dictionary:
‘Dochi’: {‘addr:housenumber’: ‘604’, ‘cuisine’: ‘donut’, ‘amenity’: ‘cafe’}

Natural language:
a cafe selling donut named Dochi

Landmark dictionary:
‘AVA Theater District’: {‘addr:housenumber’: ‘45’, ‘building’: ‘residential’,
‘building:levels’: ‘30’}
Natural language:
AVA residential building

Landmark dictionary:
‘HI Boston’: {‘operator’: ‘Hosteling International’, ‘smoking’: ‘no’,
‘wheelchair’: ‘yes’, ‘tourism’: ‘hostel’}
Natural language:
HI Boston
```

3.9.4 Implementation Details about Referring Expression Recognition

The prompt for referring expression recognition is as follows,

Referring Expression Recognition Prompt

```
Your task is to repeat exact strings from the given utterance which possibly refer
to certain propositions.

Utterance: move to red room
Propositions: red room

Utterance: visit Cutler Majestic Theater
Propositions: Cutler Majestic Theater

Utterance: robot c move to big red room and then move to green area
Propositions: big red room | green area

Utterance: you have to visit Panera Bread on Beacon Street, four or more than
four times Propositions: Panera Bread on Beacon Street

Utterance: go to Cutler Majestic Theater at Emerson College on Tremont Street,
exactly three times
Propositions: Cutler Majestic Theater at Emerson College on Tremont Street

...

Utterance: make sure you never visit St. James Church, a Christian place of
worship on Harrison Avenue, Dunkin' Donuts, Thai restaurant Montien, New Saigon
Sandwich, or Stuart St @ Tremont St
Propositions: St. James Church, a Christian place of worship on Harrison Avenue |
Dunkin' Donuts | Thai restaurant Montien | New Saigon Sandwich | Stuart St @ Tremont
St

Utterance: move the robot through yellow region or small red room and then to
large green room
Propositions: yellow region | small red room | large green room
```

```
Utterance:
```

The results of using this prompt to recognize referring expressions with spatial relations are shown in Table 3.8.

3.9.5 Implementation Details about Lifted Translation

Fintuned T5-Base

For finetuning the T5-Base model, we set the batch size to 40, the learning rate to 10^{-4}, and the weight decay to 10^{-2}. We ran training for 10 epochs and picked the best-performing one for reporting results.

Finetuned GPT-3

The per specification type accuracies and the accuracies for varying number of propositions in the formula while testing the finetuned GPT-3 model on the utterance holdout is depicted in Figure 3.3. The finetuned GPT-3 model achieves high accuracies across formula types and varying numbers of propositions. It shows the benefit of having a large high-quality dataset of natural language commands representing diverse LTL formulas. All previous works also used utterance holdout as their testing methodology, but their training and test sets contain significantly fewer unique LTL formulas.

Prompt GPT-4

The prompt for end-to-end GPT-4 is as follows,

Lifted Translation Prompt

```
Your task is to translate English utterances into linear temporal logic (LTL)
formulas.
Utterance:  visit b
LTL: F b
```

```
Utterance:  eventually reach b and h
LTL: & F b F h
Utterance:  go to h a and b
LTL: & F h & F a F b
Utterance:  proceed to reach h at the next time instant only and only if you see b
LTL: G e b X h
Utterance:  wait at b till you see h
LTL: U b h
Utterance:  go to h in the very next time instant whenever you see b
LTL: G i b X h
Utterance:
```

Prompt GPT-3

The prompt for end-to-end GPT-3 is the same as the one we used for Prompt GPT-4.

Seq2Seq Transformer

We constructed and trained a transformer model following [100]. More specifically, we built the model's encoder with three attention layers and decoder with three layers, and we used 512 as the embedding size and 8 as the number of attention heads. For training, we adapted batched training with a batch size equal to 128, learning rate equal to 10^{-4}, and dropout ratio equal to 0.1; the training process runs for 10 epochs, and we picked the best-performing checkpoint for baseline comparison.

Type Constrained Decoding (TCD)

Constrained decoding has been used in generating formal specifications for eliminating syntactically invalid outputs. Due to the sampling nature of NN-based models, generated tokens from the output layer can result in syntactical errors that can be detected on the fly, and type-constrained decoding solves it by forcing the model to only generate tokens following the correct grammar rule. By eliminating syntax errors, it also improves the

overall performance of the system.

In practice, type-constrained decoding is implemented at each step of the decoding loop: first checking the validity of the output token, then appending the valid token or masking the invalid, and re-generating a new token according to the probability distribution after masking. In addition, we design an algorithm to simultaneously enforce the length limitation and syntactical rule by parsing partial formulas into binary trees. Beyond a given maximum height of the tree, the model is forced only to generate propositions but not operators.

3.9.6 Implementation Details about Code as Policies

We designed two prompts for reproducing Code as Policies: one for code generation and the other for parsing landmarks. The code generation prompt is expected to generate an executable Python script that calls the goto_loc() function for traversing through the environment and psrse_loc() function to ground referring expressions to landmarks, where the landmark resolution prompt is used. The code generation prompt for graph search is as follows,

Code as Policies Prompt

```
# Python 2D robot navigation script


import random
from utils import goto_loc, parse_loc


# make the robot go to wooden desk.
target_loc = parse_loc(‘wooden desk’)
goto_loc(target_loc)
# go to brown desk and then white desk.
target_loc_1 = parse_loc(‘brown desk’)
target_loc_2 = parse_loc(‘white desk’)
```

```
target_locs = [target_loc_1, target_loc_2]
for target_loc in target_locs:
  goto_loc(target_loc)
# head to doorway, but visit white kitchen counter before that.
target_loc_1 = parse_loc(‘white kitchen counter’)
target_loc_2 = parse_loc(‘doorway’)
target_locs = [target_loc_1, target_loc_2]
for target_loc in target_locs:
  goto_loc(target_loc)
# avoid white table while going to grey door.
target_loc = parse_loc(‘grey door’)
avoid_loc = parse_loc(‘white table’)
target_locs = [target_loc]
avoid_locs = [avoid_loc]
for loc in target_locs:
  goto_loc(loc, avoid_locs=avoid_locs)
# either go to steel gate or doorway
target_loc_1 = parse_loc(‘steel gate’)
target_loc_2 = parse_loc(‘doorway’)
target_locs = [target_loc_1, target_loc_2]
target_loc = random.choice(target_locs)
goto_loc(target_loc)

...

# go to doorway three times
target_loc = parse_loc(‘doorway’)
for _ in range(3):
  goto_loc(target_loc)
  random_loc = target_loc
  while random_loc == target_loc:
    random_loc = random.choice(locations)
  goto_loc(random_loc)
```

The landmark resolution prompt is as follows,

Landmark Resolution Prompt

```
# Python parsing phrases to locations script


locations = [‘bookshelf’, ‘desk A’, ‘table’, ‘desk B’, ‘doorway’, ‘kitchen
counter’, ‘couch’, ‘door’]
semantic_info = {
  ‘‘bookshelf": {‘‘material": ‘‘wood", ‘‘color": ‘‘brown"},
  ‘‘desk A": {‘‘material": ‘‘wood", ‘‘color": ‘‘brown"},
  ‘‘desk B": {‘‘material": ‘‘metal", ‘color’: ‘‘white"},
  ‘‘doorway": {},
  ‘‘kitchen counter": {‘‘color": ‘‘white"},
  ‘‘couch": {‘‘color": ‘‘blue", ‘‘brand": ‘‘IKEA"},
  ‘‘door": {‘‘material": ‘‘steel", ‘‘color": ‘‘grey"},
  ‘‘table": {‘‘color": ‘‘white"},
  }
# wooden brown bookshelf
ret_val = ‘bookshelf’
...
locations = [‘bookshelf’, ‘desk A’, ‘table’, ‘desk B’, ‘doorway’, ‘kitchen
counter’, ‘couch’, ‘door’]
semantic_info = {
  ‘‘bookshelf": {‘‘material": ‘‘wood", ‘‘color": ‘‘brown"},
  ‘‘desk A": {‘‘material": ‘‘wood", ‘‘color": ‘‘brown"},
  ‘‘desk B": {‘‘material": ‘‘metal", ‘color’: ‘‘white"},
  ‘‘doorway": {},
  ‘‘kitchen counter": {‘‘color": ‘‘white"},
  ‘‘couch": {‘‘color": ‘‘blue", ‘‘brand": ‘‘IKEA"},
  ‘‘door": {‘‘material": ‘‘steel", ‘‘color": ‘‘grey"},
```

```
  ‘‘table": {‘‘color": ‘‘white"},
  }
# blue IKEA couch
ret_val = ‘couch’
```

3.9.7 Implementation Details about Grounded Translation

CopyNet

For reproducing [127], we trained the CopyNet baseline with our grounded dataset preprocessed as its required format. To make a fair comparison on generalization ability, the CopyNet model has only seen utterance-formula pairs from the Boston subset, and the evaluation is run on grounded datasets of the rest 21 cities. For training CopyNet, we followed closely the instructions of the original paper and used the exact same LSTM model structure and pre-computed glove embedding for landmark resolution. On the hyperparameters, we set the embedding size to 128, the hidden size to 256, the learning rate to 10^{-3}, and the batch size to 100.

Prompt GPT-4

The prompt for end-to-end GPT-4 is as follows. While we tried including a landmark list in the prompt, it was removed in the final version because we observed empirically that Prompt GPT-4 achieved better performance without explicitly giving a list of landmarks during prompt engineering.

Grounded Translation Prompt

```
Your task is to first find referred landmarks from a given list then use them
as propositions to translate English utterances to linear temporal logic (LTL)
formulas.
Utterance:  visit Panera Bread sandwich fast food on Stuart Street
LTL: F panera_bread
Utterance:  eventually reach Wang Theater, and The Kensington apartments
```

```
LTL: & F wang_theater F the_kensington
...
Utterance:  make sure that you have exactly three separate visits to Seybolt Park
LTL: M & seybolt_park F & !  seybolt_park F & seybolt_park F & !  seybolt_park F
seybolt_park | !  seybolt_park G | seybolt_park G | !  seybolt_park G | seybolt_park
G | !  seybolt_park G | seybolt_park G ! seybolt_park
Utterance:
```

3.9.8 Dataset Details

Quantifying diversity of temporal commands

We quantify the diversity of the temporal commands a system is tested on using the temporal formula skeletons in the evaluation corpus of commands. We propose that each novel dataset should be characterized along the following dimensions, and as an example, we provide the respective values for the Lang2LTL dataset (lifted and grounded OSM dataset) described in Section 6.4 of the main paper.

1. Number of semantically unique formulas: 47
2. Number of propositions per formula: minimum: 1, maximum: 5, average: 3.79
3. Length of formulas: minimum: 2, maximum: 67. average: 18.89
4. Vocabulary size (for grounded datasets): 1757
5. Linguistic diversity of utterances: self-BLEU score: 0.85

Table 3.4: Dataset Comparison

	Lang2LTL Lifted	CleanUp World	NL2TL	Wang et al. [35]
Number of datapoints	49,655	3,382	39,367	6,556
Unique formula skeletons	47	4	605	45
#Propositions (min, max, mean)	(1, 5, 3.79)	(1, 3, 1.85)	(1, 7, 2.86)	(1, 4, 2.01)
Formula Length (min, max, mean)	(2, 67, 18.89)	(2, 7, 4.77)	(1, 13, 5.98)	(3, 7, 4.48)

Table 3.4 compares our proposed lifted dataset and other datasets proposed in prior

work.

3.9.9 Detailed Result Analysis on Lifted Translation

We further analyzed the results for each model and holdout type for the lifted translation problem. In particular, we computed the accuracies per each formula type and the number of unique propositions required to construct the target formula. This analysis provides insights into the sensitivity of the models to particular templates and formula lengths.

The accuracies of each model and holdout type categorized by formula types are depicted in Figure 3.4. We observe that for both the finetuned models (Finetuned T5 and Finetuned GPT-3), the model achieves high accuracies across various formula types for *Utterance Holdout*. Note that the performance across types is more uniform for the Finetuned GPT-3 than the Finetuned T5-Base model. Next, we note that Prompt GPT-4 achieves better accuracies as compared to Prompt GPT-3 across all evaluations.

We observe that the performance of the finetuned models is more unbalanced across different formula types for the *Formula Holdout* test case. In comparison, Prompt GPT models achieve non-zero accuracies across all formula types. Once again, Prompt GPT-4 outperforms Prompt GPT-3. We note that adding type-constrained decoding to Finetuned T5-Base during inference only marginally improved *Utterance Holdout*, but significantly improved *Formula* and *Type Holdout*, which implies Finetuned T5-Base model is more likely to produce syntactically incorrect output when the grounding formula instance or type have not seen during training.

Finally, we note that only the prompt GPT models achieve meaningful accuracies in the *Type Holdout* scenarios. However, even in *Type Holdout*, the accuracies are concentrated on formula types that only had short lengths or shared subformulas with types seen during training. We can conclude that *Formula* and *Type Holdout* remain challenging paradigms of generation and an open problem for automated translation of language commands into formal specifications.

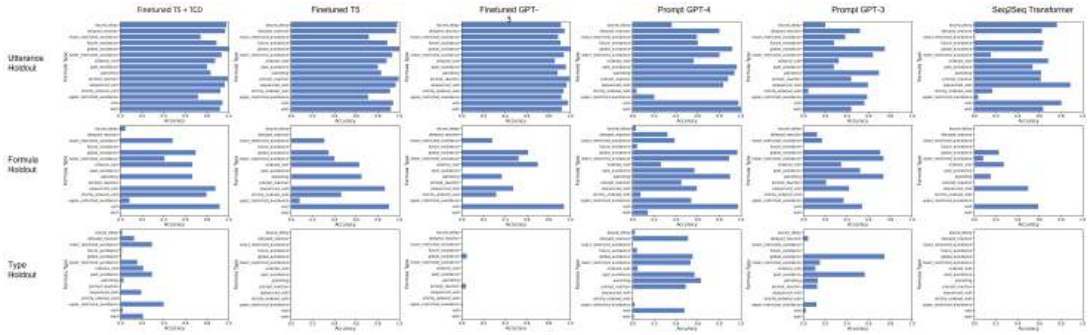


Figure 3.4: The accuracies per grounding formula types of six lifted translation models

Next, we repeated the above analysis but categorized accuracies by the number of unique propositions that appear within a formula. The results are depicted in Figure 3.5.

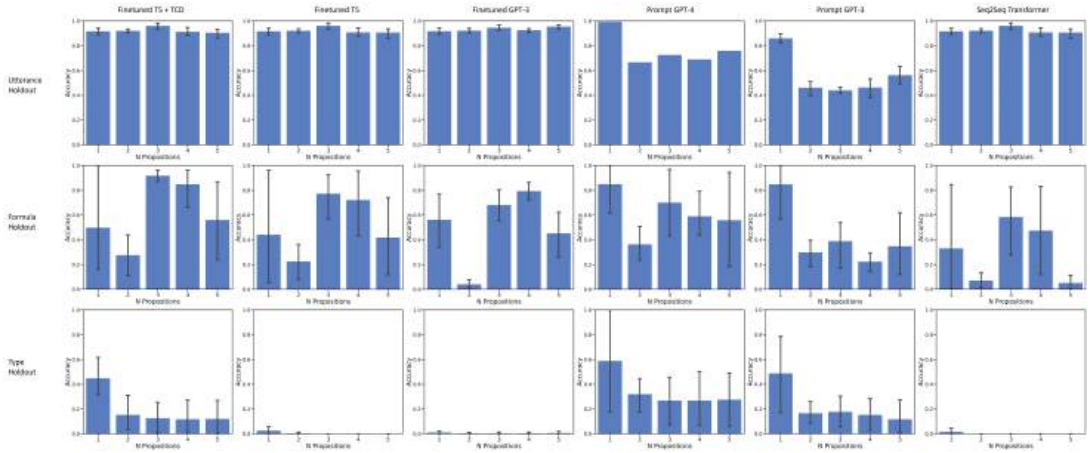


Figure 3.5: The accuracies per number of unique propositions of six lifted translation models

Here we note that in the *Utterance Holdout* test, the finetuned models demonstrated balanced performance across the dataset, whereas both prompt GPT models demonstrated degraded performance when the number of unique propositions in the formula was increasing. Subsequently, the degraded performance on longer formulas was apparent even within the *Formula* and *Type holdout* domains. In contrast, the three finetuned models performed better for longer formulas in *Formula Holdout*. We hypothesize that this is because the finetuned models were more able to generalize to different formula lengths of the same template (in particular, the templates that required temporal ordering constraints to be encoded) as compared to the prompt completion-based approaches. In addition,

there are more samples in the training set for longer formulas due to permutations of propositions.

As finetuning an LLM on the target task produced the best results for *Utterance Holdout*, we further analyzed the cause of errors for the instances where the lifted translation was incorrect. We categorize the errors as follows:

1. **Syntax Errors:** The formula returned by the lifted translation module was not a valid LTL formula.
2. **Misclassifed formula type:** The lifted translation module returns an identifiable but incorrect formula type that did not correspond to the input command.
3. **Incorrect propositions:** The returned formula was of the correct formula type but had the incorrect number of propositions.
4. **Incorrect permutation:** The formula was of the correct template class and had the right number of propositions, but the propositions were in the wrong location within the formula.
5. **Unknown template** The returned formula was a valid LTL formula but did not belong to any known formula types.

Figure 3.6 to Figure 3.8 depict the relative frequencies of the error cases as a pie chart for the three finetuned models. Note that returning unknown formula templates with the correct syntax was the most common cause of error in the lifted translation based on all finetuned models.

Since Finetuned GPT-3 achieves the best generalization across formula types, and type-constrained decoding (TCD) during inference significantly improves the translation accuracies for unseen formula instances and types, the combination of large language models and TCD is by far the best approach for grounding language commands for temporal tasks.

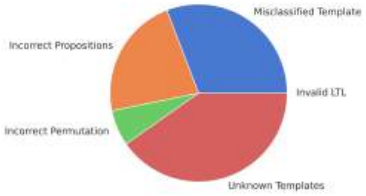


Figure 3.6: Error frequencies of Finetuned T5-Base with TCD

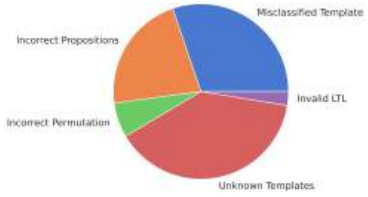


Figure 3.7: Error frequencies of Finetuned T5-Base

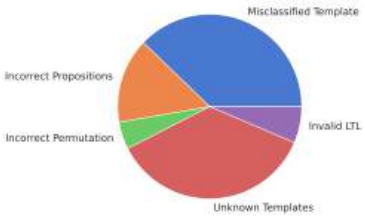
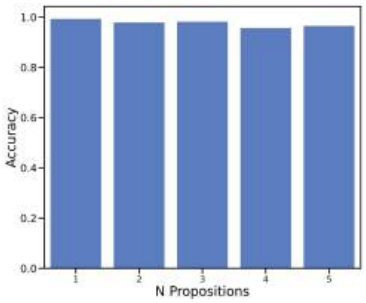
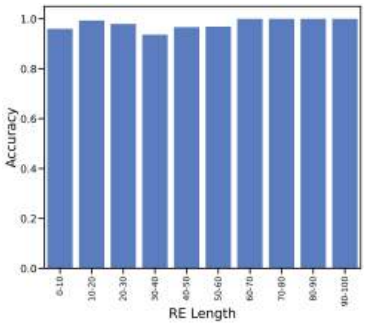


Figure 3.8: Error frequencies of Finetuned GPT-3



(a) RER Accuracy vs. Command Complexity



(b) REG Accuracy vs. RE Complexity

Figure 3.9: Figure 3.9a shows the accuracies of the referring expression recognition (RER) module as the complexity of commands (measured by the number of referring expressions in the command) increases. Figure 3.9b shows the accuracy of the referring expression grounding (REG) module as the complexity of REs (measured by string length) increases.

3.9.10 Robot Demonstration

Indoor Environment #1

The semantic information of landmarks in the first household environment is as follows,

```
{
    "bookshelf": {
        "material": "wood",
        "color": "brown"
    },
    "desk A":{
        "material": "wood",
        "color": "brown"
```

```
    },
    "desk B": {
        "material": "metal",
        "color": "white"
    },
    "doorway": {},
    "kitchen counter": {
        "color": "white"
    },
    "couch": {
        "color": "blue",
        "brand": "IKEA"
    },
    "door": {
        "material": "steel",
        "color": "grey"
    },
    "table": {
        "color": "white"
    }
}
```

Natural language commands used to test our system Lang2LTL and Code as Polices [8], are shown in Table 3.5.

Indoor Environment #2

The semantic information of landmarks in the second household environment is as follows,

```
{
```

```
    "hallway A": {
        "decoration": "painting"
    },
    "hallway B": {
        "decoration": "none"
    },
    "table A": {
        "location": "kitchen",
        "material": "metal",
        "color": "blue"
    },
    "table B": {
        "location": "atrium",
        "material": "metal",
        "color": "white"
    },
    "classroom": {
        "door": ["glass", "grey"]
    },
    "elevator": {
        "color": "purple"
    },
    "staircase": {},
    "front desk": {},
    "office": {
        "door": ["wood", "yellow"]
    },
}
```

Natural language commands used to test our system Lang2LTL and Code as Polices [8], are shown in Table 3.6.

Table 3.5: Commands for Robot Demonstration in Indoor Environment #1

Navigational Command	Lang2LTL Result	Code as Policies Result
1. go to brown bookshelf, metal desk, wooden desk, kitchen counter, and the blue couch in any order	success	success
2. move to grey door, then bookrack, then brown desk, then counter, then white desk	success	success
3. visit brown wooden desk but only after bookshelf	success	misunderstand the task
4. go from brown bookshelf to white metal desk and only visit each landmark one time	success	misunderstand the task
5. go to brown wooden desk exactly once and do not visit brown desk before bookshelf	success	inexecutable
6. go to white desk at least three times	success	inexecutable
7. go to wooden bookshelf at least five times	success	success
8. visit bookshelf at most three times	success	success
9. visit counter at most 5 times	success	success
10. go to wooden desk exactly three times	success	misunderstand the task
11. move to brown wooden desk exactly 5 times	success	inexecutable
12. go to doorway exactly two times, in addition always avoid the table	success	success
13. go to brown desk only after visiting bookshelf, in addition go to brown desk only after visiting white desk	success	misunderstand the task
14. visit wooden desk exactly two times, in addition do not go to wooden desk before bookrack	success	inexecutable
15. visit wooden desk at least two times, in addition do not go to wooden desk before bookshelf	success	inexecutable
16. visit the blue IKEA couch, in addition never go to the big steel door	success	success
17. visit white kitchen counter then go to brown desk, in addition never visit white table	success	success
18. go to the grey door, and only then go to the bookshelf, in addition always avoid the table	success	misunderstand the task
19. go to kitchen counter then wooden desk, in addition after going to counter, you must avoid white table	success	misunderstand the task
20. Go to bookshelf, alternatively go to metal desk	success	misunderstand the task
21. Go to counter, alternatively go to metal desk	success	misunderstand the task
22. Go to the counter, but never visit the counter	unsatisfiable. abort correctly	stop execution correctly
23. do not go to the wooden desk until bookshelf, and do not go to bookshelf until wooden desk	unsatisfiable. abort correctly	stop execution correctly
24. go to brown desk exactly once, in addition go to brown desk at least twice	unsatisfiable. abort correctly	misunderstand the task
25. find the kitchen counter, in addition avoid the doorway	unsatisfiable. abort correctly	stop execution correctly
26. move to couch exactly twice, in addition pass by counter at most once	unsatisfiable. abort correctly	stop execution correctly
27. navigate to the counter then the brown desk, in addition after going to the counter, you must avoid doorway	unsatisfiable. abort correctly	misunderstand the task
28. Visit the counter at least 2 times and at most 5 times	incorrect grounding. OOD	inexecutable
29. visit counter at least six times	incorrect grounding. OOD	success
30. either go to bookshelf then desk A, or go to couch	incorrect grounding. OOD	misunderstand the task

Table 3.6: Commands for Robot Demonstration in Indoor Environment #2

Navigational Command	Lang2LTL Result	Code as Policies Result
1. navigate to the office with the wooden door, the classroom with glass door and the table in the atrium, kitchen counter, and the blue couch in any order	success	success
2. go down the hallway decorated with paintings, then find the kitchen table, then front desk, then staircase	success	success
3. navigate to classroom but do not visit classroom before the white table in atrium	success	misunderstand the task
4. only visit classroom once, and do not visit classroom until you visit elevator first	success	success
5. Go to the staircase, front desk and the white table in the atrium in that exact order. You are not permitted to revisit any of these locations	success	inexecutable
6. go to the purple elevator at least five times	success	inexecutable
7. visit the kitchen table at most three times	success	success
8. navigate to the classroom exactly four times	success	inexecutable
9. go to the front desk then the yellow office door, in addition do not visit the classroom with glass door	success	success
10. go to the stairs then the front desk, in addition avoid purple elevator	success	success
11. move to elevator then front desk, in addition avoid staircase	success	success
12. go to front desk exactly two times, in addition avoid elevator	success	inexecutable
13. Go to elevator, alternatively go to staircase	success	misunderstand the task
14. Go to the front desk at least two different occasions, in addition you are only permitted to visit the staircase at most once	success	misunderstand the task
15. Visit the elevator exactly once, in addition visit the front desk on at least 2 separate occasions	success	inexecutable
16. Go to the office, in addition avoid visiting the elevator and the classroom	success	success
17. Visit the front desk, in addition you are not permitted to visit elevator and staircase	success	success
18. Visit the purple door elevator, then go to the front desk and then go to the kitchen table, in addition you can never go to the elevator once you've seen the front desk	success	inexecutable
19. Visit the front desk then the white table, in addition if you visit the staircase you must avoid the elevator after that	success	inexecutable
20. Go to the classroom with glass door, but never visit the classroom with glass door	unsatisfiable. abort correctly	stop execution correctly
21. do not go to the white table until classroom, and do not go to the classroom until white table	unsatisfiable. abort correctly	stop execution correctly
22. go to kitchen table exactly once, in addition go to kitchen table at least twice	unsatisfiable. abort correctly	misunderstand the task
23. find the office, in addition avoid visiting the front desk and the classroom and the table in atrium	unsatisfiable. abort correctly	stop execution correctly
24. move to the kitchen table exactly twice, in addition pass by hallway decorated by paintings at most once	unsatisfiable. abort correctly	misunderstand the task
25. navigate to the kitchen table then the front desk, in addition after going to the kitchen table, you must avoid hallway decorated with paintings	unsatisfiable. abort correctly	misunderstand the task
26. Go to the front desk at least 4 different occasions, additionally, you are only permitted to visit the staircase at most once	incorrect grounding. OOD	inexecutable
27. Visit the front desk, additionally if you visit the elevator you must visit the office after that	incorrect grounding. OOD	success
28. Visit the front desk, additionally you visit the elevator you must visit the office after that the white table and the classroom	incorrect grounding. OOD	misunderstand the task

Table 3.7: Example Commands from OpenStreetMap Dataset

LTL Type	Command (with two referring expressions)
Visit	move to Thai hot pot restaurant on Kneeland Street, and Vietnamese restaurant on Washington Street
Sequence Visit	visit Subway sandwich shop on The Plaza, followed by Zada Jane's Cafe on Central Avenue
Ordered Visit	find Local Goods Chicago gift shop, but not until you find Currency exchange bureau, first
Strictly Ordered Visit	reach Citibank branch, and then Cutler Majestic Theater on Tremont Street, in that exact order without repetitions
Patrolling	keep on visiting US Post Office on West Devon Avenue, and Kanellos Shoe Repair shop
Bound Delay	you must go to Purple Lot parking area, immediately after you visit Royal Nails & Spa on South Main Street, and you can not go to Purple Lot parking area, any other time
Delayed Reaction	you must visit Peruvian restaurant on Virginia Avenue, once you visit PNC Bank
Prompt Reaction	immediately after you go to Beachside Resortwear clothing store, you must go to Walgreens Pharmacy
Wait	you can not go to other place from Publix supermarket, unless you see Beaches Museum
Past Avoidance	avoid visiting IES Test Prep school, till you observe bookstore on Elizabeth Street
Future Avoidance	never go to Commercial building on 5th Avenue, once you go to Cafe Metro
Global Avoidance	make sure to never reach either Citibank, or Seybolt Park
Upper Restricted Avoidance	go to Cocktail bar, at most twice
Lower Restricted Avoidance	you have to visit Main Branch of CoGo Bike Share Library for bicycle rental, two or more than two times
Exact Restricted Avoidance	navigate to Art shop on Bannock Street, exactly twice

Table 3.8: Results of Recognizing Referring Expression with Spatial Relations

Navigational Command	Referring Expression(s)	Correctness
1. go to back of Common Market	back of Common Market	correct
2. always avoid entrance and exit of Little Sugar Creek, but visit left and right of Little Sugar Creek	entrance and exit of Little Sugar Creek | left and right of Little Sugar Creek	correct
3. stay at intersection of Thayer street and Waterman street	intersection of Thayer street and Waterman street	correct
4. move forward to the south of Edgebrook Coffee Ship	south of Edgebrook Coffee Ship	correct
5. go to east of Chinatown, without visiting west of New Saigon Sandwich, then go to front of New Saigon Sandwich, without visiting rear of Dumpling Cafe, then go to rear of Dumpling Cafe, without visiting north of Emerson College - Little Building, finally go to south Emerson College - Little Building, while only visiting each location once	east of Chinatown | west of New Saigon Sandwich | front of New Saigon Sandwich — rear of Dumpling Cafe | rear of Dumpling Cafe | north of Emerson College - Little Building | south Emerson College - Little Building	correct
6. go around big blue box	big blue box	incorrect
7. go to exit of blue area through between red room and blue one	exit of blue area | red room | blue one	incorrect
8. go to left of CVS and the stay on bridge	left of CVS | bridge	incorrect
9. go pass right of Dairy Queen to left of Harris Teeter, end up at entrance of Wells Fargo	Dairy Queen | Harris Teeter | Wells Fargo	incorrect
10. move to My Sister's Closet and stop close to bus stop near Ace Hardware	My Sister's Closet | bus stop near Ace Hardware	incorrect

CHAPTER 4

Grounding Spatiotemporal Language

In the previous Chapter 3, we introduced a modular system that can ground natural language that specifies temporally-extended commands. However, many natural language instructions from humans contain spatial constraints in addition to the temporal ones. Grounding spatiotemporal navigation commands to structured task specifications enables autonomous robots to understand a broad range of natural language and solve long-horizon tasks with safety guarantees. Prior works mostly focus on grounding spatial or temporally extended language for robots. We propose Lang2LTL-2, a modular system that leverages pretrained large language and vision-language models and multimodal semantic information to ground spatiotemporal navigation commands in novel city-scaled environments without retraining. Lang2LTL-2 achieves 93.53% language grounding accuracy on a dataset of 21,780 semantically diverse natural language commands in unseen environments. We run an ablation study to validate the need for different modalities. We also show that a physical robot equipped with the same system without modification can execute 50 semantically diverse natural language commands in both indoor and outdoor environments.

4.1 Introduction

When giving directions, humans often use natural language that describes goals, as well as temporal and spatial constraints. For example, consider the command "Visit the Starbucks, only then go to the red car to the right of the building, and always avoid the crowded restaurant near the cafe." An autonomous robot following this spatiotemporal command must understand that it specifies a temporally extended task of visiting two locations in a strict order while avoiding the third throughout the execution. The robot must ground the three referring expressions, i.e., "the Starbucks," "the car," and "the crowded restaurant," to specific locations with respect to other landmarks in the environment.

Existing approaches focus on developing the robot's spatial or temporal reasoning ability separately. Many works develop systems to ground natural language commands that contain rich spatial relations in indoor [133, 134, 135] and outdoor [136, 137] environments. Their approaches use a map that contains multimodal semantic information to identify various target landmarks with respect to others in the environment, yet they cannot handle complex temporal constraints. Separately, structured task specifications, like linear temporal logic (LTL), can capture a wide range of semantically diverse temporal patterns [113] and enable the synthesis of verifiable robot behaviors with safety guarantees. However, systems that can ground complex temporal language have limited spatial reasoning capability [40, 73, 138].

To achieve the best of both worlds, we introduce a modular language grounding system, Lang2LTL-2, that grounds spatiotemporal navigation commands for robots. Lang2LTL-2 uses large language models (LLMs) to recognize spatial referring expressions, like "the red car to the right of the building," and to translate language commands to LTL task specifications, which are compatible with many planning and reinforcement learning algorithms [139, 140, 141, 142, 143]. Using pretrained vision-language models (VLMs) and text embedding, Lang2LTL-2 grounds referring expressions to specific locations in

Figure 4.1: Our system Lang2LTL-2 grounds spatiotemporal navigation commands in indoor and outdoor environments. The spatial and temporal components of the example commands are highlighted in blue and red, respectively.

novel city-scaled environments without retraining, given a semantic database of textual and visual descriptions of the landmarks.

We evaluated Lang2LTL-2 on a dataset of 21,780 semantically diverse spatiotemporal commands with 1,723 spatial referring expressions, 19 spatial relations, and 15 temporal patterns. We also ran an ablation study and showed that using multimodal semantic information for spatiotemporal language grounding outperforms using any modality alone. Finally, we demonstrated that a mobile robot equipped with the same system without modification could execute 50 semantically diverse spatiotemporal commands in both indoor and outdoor environments. Code, datasets, and videos are at `spatiotemporal-ground.github.io`.

4.2 Preliminaries

4.2.1 Large Language Models and Vision-Language Models

Large language models (LLMs) are transformer neural networks [100] trained to maximize the probability of a successive token given a context window. They achieve state-of-the-art (SoTA) performance on a wide variety of natural language processing tasks [102]. Pretrained LLMs can also produce high-dimensional embedding vectors of text. We can measure the semantic similarity of two pieces of text by computing the

cosine similarity of their embeddings. In this work, we used OpenAI's GPT-4 model [104] and embedding API for text completion and text embedding, respectively, and a fine-tuned T5-base model [144] to translate natural language commands to temporal task specification.

Vision-language models (VLMs) are multimodal models jointly trained on text and images [145]. They produce SoTA results on many language-conditioned vision tasks [146], e.g., open-vocabulary object detection [147, 148], image captioning [149], image retrieval [150], and visual question answering [151]. In this work, we prompted the GPT-4V(ision) model [152] to generate captions for images of landmarks and objects.

4.2.2 Temporal Task Specification

Linear temporal logic (LTL) [88] is a promising task specification language for human-centered specification elicitation [73, 108, 153, 154], planning [108, 143], and reinforcement learning [139, 155]. The syntax of LTL is defined through the following recursive grammar:

$$\varphi := \alpha \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \mathbf{X}\varphi \mid \varphi_1 \ \mathbf{U} \ \varphi_2 \tag{4.1}$$

Here α represents an atomic Boolean proposition, and $\varphi,\ \varphi_1,\ \varphi_2$ are any valid LTL formulas. The operators $\neg$ (not) and $\vee$ (or) are identical to propositional logic operators. The formula $\mathbf{X}\varphi$ holds if φ holds at the next time step, and $\varphi_1 \ \mathbf{U} \ \varphi_2$ holds if φ_1 holds at least until φ_2 first holds, which must happen at the current or a future time. LTL syntax also admits abbreviated operators defined through the compositions of the primitive operators. In this work, we use the operators $\wedge$ (and), $\mathbf{F}$ (read "finally" or "eventually"), and $\mathbf{G}$ (read "globally" or "always"). $\mathbf{F}\varphi$ specifies that the formula φ must hold at least once in the future, and $\mathbf{G}\varphi$ specifies that φ must always hold.

4.2.3 Task Execution for Temporal Task Specification

Our language grounding system Lang2LTL-2 is compatible with many planning and reinforcement learning algorithms that solve LTL tasks [139, 140, 141, 142, 143, 156]. We can transform LTL formulas to Büchi automata [114, 115]. Transitions in the environment induce transitions in the automaton, so we can track task progress by tracking the automaton's state transition. We can then compute a policy on the product MDP of the task automaton and the environment MDP.

4.3 Problem Definition

Our language grounding system Lang2LTL-2 receives a natural language utterance u from the user that specifies a navigation task in an environment modeled as $\langle \mathcal{S}, \mathcal{A}, T \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ represent the robot's states and actions, and $T(s, a) \rightarrow s'$ captures the transition dynamics. In this work, we consider navigational actions that transition a robot from one location to another in the environment represented as a semantic map. We assume the robot has access to a multimodal semantic database $\mathcal{D} = \{p : (d, f)\}$, where p is a proposition that uniquely represents a landmark in the environment, d is a semantic description of the landmark, and $f : \mathcal{S} \rightarrow \{0, 1\}$ is a Boolean-valued function that determines the true value of the proposition in a given state. The semantic description of a landmark can be a piece of text (including its name, amenity, street address, etc.), an image, or both. Lang2LTL-2 translates the input command to a linear temporal logic (LTL) formula φ and grounds its propositions to landmarks in the real world. We assume the robot can track its state in a semantic map and has access to an automated planner that, given an LTL formula as task specification, produces a trajectory in the semantic map. Many planning and reinforcement learning algorithms [139, 140, 141, 142, 143] are compatible with LTL task specification. We use the AP-MDP planner [141]. Figure 4.2 shows an example execution by the full system, i.e., language grounding and planning.

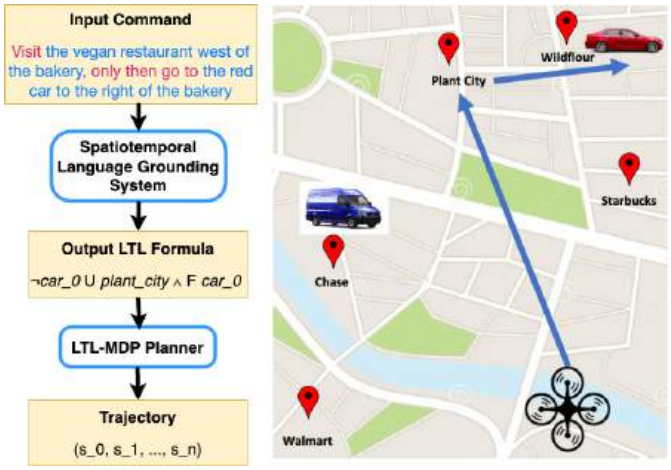


Figure 4.2: An example of an input spatiotemporal navigation command whose spatial and temporal components are highlighted in blue and red, respectively, an output LTL formula whose propositions are grounded to physical landmarks, and an execution trajectory in the environment.

4.4 Lang2LTL-2: Spatiotemporal Language Grounding

We approach the problem of spatiotemporal language grounding with a modular design, where we extract spatial referring expressions and translate temporal commands using large language models, and ground referring expressions to physical landmarks using a vision-language model and text embedding. Our system Lang2LTL-2 produces a grounded temporal task specification with grounded referring expressions and spatial relations. Figure 4.3 shows an overview of our language grounding system.

4.4.1 Spatial Referring Expression Recognition (SRER)

The spatial referring expression recognition (SRER) module identifies spatial referring expressions in a given language command. Referring expressions (REs) are noun phrases, pronouns, and proper names that refer to some entity in an environment, such as landmarks and objects [125]. In this work, we only consider noun phrases and proper names and leave the coreference resolution problem to future work. Spatial referring expressions (SREs) are phrases where referring expressions are connected by a spatial relation. For

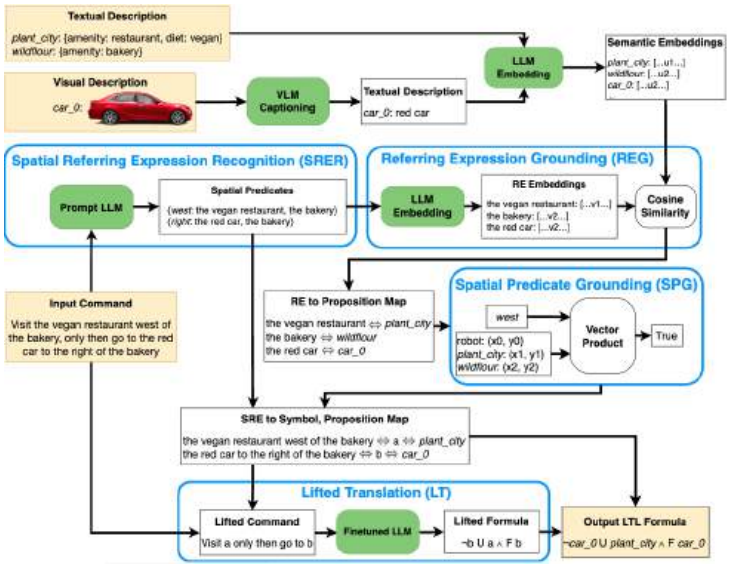


Figure 4.3: Lang2LTL-2 Language Grounding System Overview: input and output are in yellow blocks; modules are in blue blocks; pretrained and fine-tuned models are in green blocks.

example, in the language command "Go to the red car to the right of the bakery," the SRE "the red car to the right of the bakery" contains two REs, "the red car" and "the bakery," termed the figure e_f and the ground e_g, respectively, by Landau and Jackendoff [157]. The figure e_f and the ground e_g are connected by the spatial relation r "to the right of." We define a diverse set $\mathcal{R}$ of 19 spatial relations, such as near, in front of, behind, to the left of, to the right of, between, and four cardinal directions. The SRER module extracts referring expressions and their spatial relations from a spatiotemporal language command by prompting an LLM. We use GPT-4 [104]. The output of the SRER module is a spatial predicate denoted by $\{r : (e_f, e_g)\}$. For the complete set of spatial relations and the prompt used for SRER, please see the Additional Details sections 4.9.1 and 4.9.2, respectively.

4.4.2 Referring Expression Grounding (REG)

To ground the referring expressions (REs) e_f and e_g to physical landmarks in the environment, we use a multimodal semantic database with textual and visual descriptions of landmarks. Having both modalities enables the referring expression grounding (REG)

module to ground more complex REs and improves grounding accuracy. For certain REs, one modality is more descriptive than the other. For example, a textual description including a landmark's amenity and cuisine type better matches the RE "the vegan restaurant" than an image of the restaurant front. The REG module is important for identifying possible candidates for each RE, especially when the environment contains multiple similar landmarks or objects.

In practice, detailed textual descriptions of objects are not always available, e.g., "the red car," but can be extracted from images. We prompt a pretrained vision-language model (VLM) to generate captions of images with the question, "What is the most obvious object in this image?" In this work, we use GPT-4V(ision) [152]. We then use an LLM to generate text embeddings for the image captions, the textual descriptions of landmarks in the semantic database, and the query REs (i.e., e_f and e_g) extracted from the language command. Finally, we use the cosine similarity between text embeddings to find the landmarks that best match the query REs. Let $g_{caption} : i \to t$ be the function that generates a caption t for image i parameterized by the weights of the VLM, and $g_{embed} : t \to z$ be the function that computes an n-dimensional embedding z of a text string t parameterized by the weights of the LLM. The cosine similarity score is defined as follows,

$$score(e_{f/g}, t) = \frac{g_{embed}(e_{f/g})^T g_{embed}(t)}{||g_{embed}(e_{f/g})|| \cdot ||g_{embed}(t)||}, \tag{4.2}$$

where we substitute $t = g_{caption}(i)$ when the semantic description of the landmark is an image i, and $e_{f/g}$ denotes the query RE being the figure e_f or the ground e_g.

We also explored using CLIP's text and image encoders [145] to encode text and images, then the cosine similarity of the text and image embeddings to find the best matching landmark for a query RE. However, we discovered that the gap between the text and image embedding spaces is large for the pretrained CLIP model. Liang et al. [158] documented this phenomenon in more detail. Instead of training another neural network to align the text and image embedding spaces, we use a pretrained VLM to transcribe

images to text and work solely in the text embedding space.

4.4.3 Spatial Predicate Grounding (SPG)

After grounding the figure e_f and the ground e_g to candidate landmarks, we perform spatial predicate grounding (SPG) to identify the most likely landmark referred to by e_f given e_g and the spatial relation r. We assume that users give commands with respect to the robot's initial location. For each spatial referring expression (SRE) and its corresponding spatial predicate $\{r : (e_f, e_g)\}$, we rank all the candidate landmarks of e_f based on the product of the similarity scores computed by the referring expression grounding (REG) module for the candidate landmarks of e_f and e_g, then select the proposition p_f with the highest product score,

$$p_f^* = \underset{p_f:(d_f,\_)\in\mathcal{D}, p_g:(d_g,\_)\in\mathcal{D}}{\arg\max} score(e_f, d_f) \cdot score(e_g, d_g). \tag{4.3}$$

To validate each pair of candidate landmarks, we first compute a ground vector from the ground landmark to the robot, which serves as an anchor for computing the range where the figure landmark should be. We then compute a figure vector from the ground landmark to the figure landmark. Based on the spatial relation, we compute a range where the figure vector should lie relative to the ground vector. Figure 4.4 illustrates the

Figure 4.4: An illustration of the ground vector and the figure vector, depicted as the green and the red arrow, respectively, computed by the spatial predicate grounding (SPG) module (Section 4.4.3) to resolve the spatial referring expression "the red car to the right of the bakery."

ground and the figure vectors for the SRE "the red car to the right of the bakery."

For each known spatial relation $r \in \mathcal{R}$, we specify a set of rules to validate a pair of candidate landmarks for e_f and e_g. In the example of "the red car to the right of the bakery," the spatial relation "to the right of" means the figure vector must lie within the half circle between the ground vector and 180 degrees counterclockwise from the ground vector. Please see the Additional Details section 4.9.1 for the definition of all spatial relations. We also specify a distance threshold in meters between a figure and the ground to eliminate candidate figures too far from the ground. To resolve an unseen spatial relation, we use LLM text embedding and cosine similarity to find the most semantically similar spatial relation $r \in \mathcal{R}$.

4.4.4 Lifted Translation (LT)

After the SRER module extracts all the spatial referring expressions (SREs) from a given command, we transform it into a lifted command by substituting the SREs with symbols, which are grounded to physical landmarks by the REG (Section 4.4.2) and the SPG (Section 4.4.3) modules. For example, the input command "Go to the red car to the right of the bakery" is transformed to a lifted command "Go to a" where the symbol a substitutes the SRE "the red car to the right of the bakery." We then translate the lifted command to a lifted LTL formula compatible with many planning and reinforcement learning algorithms [139, 140, 141, 142, 143]. We evaluate the following models for lifted translation.

Fine-tuned LLM: Liu et al. [138] tested four models that use LLMs for lifted translation. The T5-Base (220M parameters) model [144] fine-tuned on the semantically diverse dataset they collected overperformed the fine-tuned GPT-3 [103], the Prompt GPT-3 [103] and the Prompt GPT-4 [104] models. Thus, we use their best-performing model fine-tuned T5-Base through HuggingFace's Transformer library [159].

Retrieval Augmented Generation (RAG): We evaluate retrieval augmented

generation (RAG), which dynamically constructs a prompt to an LLM based on the query [160] for lifted translation. To translate a lifted command to a lifted LTL formula with RAG, we use cosine similarity of text embeddings to find semantically similar commands from the lifted dataset collected in [138], then use these commands and their corresponding LTL formulas as in-context examples to query GPT-4 [104]. We test varying numbers of in-context examples. Please see the Additional Details Section 4.9.3 for an example prompt used for RAG.

Table 4.1: Modular Performance

Module		**Accuracy** (averaged over five seeds)				
		City 1 (9 landmarks)	*City 2* (34 landmarks)	*City 3* (44 landmarks)	*City 4* (175 landmarks)	*Average*
SRER		$99.45 \pm 0.12\%$	$99.43 \pm 0.26\%$	$99.56 \pm 0.63\%$	$99.39 \pm 0.21\%$	$99.46 \pm 0.34\%$
REG	Top-1	$99.68 \pm 0.72\%$	$97.98 \pm 1.07\%$	$88.74 \pm 2.14\%$	$78.35 \pm 1.97\%$	$91.19 \pm 8.84\%$
	Top-5	$100.00 \pm 0.00\%$	$100.00 \pm 0.00\%$	$99.56 \pm 0.24\%$	$99.15 \pm 0.34\%$	$99.68 \pm 0.41\%$
	Top-10	$100.00 \pm 0.00\%$	$100.00 \pm 0.00\%$	$99.70 \pm 0.17\%$	$99.98 \pm 0.05\%$	$99.92 \pm 0.15\%$
SPG		$100.00 \pm 0.00\%$	$100.00 \pm 0.00\%$	$99.53 \pm 0.33\%$	$99.35 \pm 1.46\%$	$99.72 \pm 0.75\%$
LT	Finetuned T5-base	$99.45 \pm 0.00\%$	$99.45 \pm 0.00\%$	$99.45 \pm 0.00\%$	$99.45 \pm 0.00\%$	$99.45 \pm 0.00\%$
	RAG-10	$69.33 \pm 0.25\%$	$70.34 \pm 0.13\%$	$69.65 \pm 0.58\%$	$70.39 \pm 0.84\%$	$69.93 \pm 0.62\%$
	RAG-50	$83.79 \pm 0.06\%$	$83.93 \pm 0.12\%$	$83.75 \pm 0.52\%$	$83.93 \pm 0.65\%$	$83.85 \pm 0.33\%$
	RAG-100	$88.20 \pm 0.58\%$	$88.25 \pm 1.04\%$	$87.79 \pm 0.39\%$	$87.70 \pm 0.13\%$	$87.98 \pm 0.54\%$

4.5 Evaluation of Language Grounding

We conducted three sets of evaluations of our spatiotemporal language grounding system Lang2LTL-2: 1) a modular evaluation, where we tested the performance of individual modules introduced in Section 4.4, 2) a full system evaluation, where we evaluated the final output of our system, and 3) an ablation study of the text and the image modality.

4.5.1 Language Grounding Dataset

Our evaluation used four city-scaled environments with an increasing number of landmarks, i.e., 9, 34, 44, and 175. The landmarks were described by text from OpenStreetMap [161] (e.g., names, street addresses, amenities, and GPS coordinates, etc.) and images from Google StreetView [162]. Having a dataset where landmarks are described by both modalities helps evaluate whether the referring expression grounding (REG) module

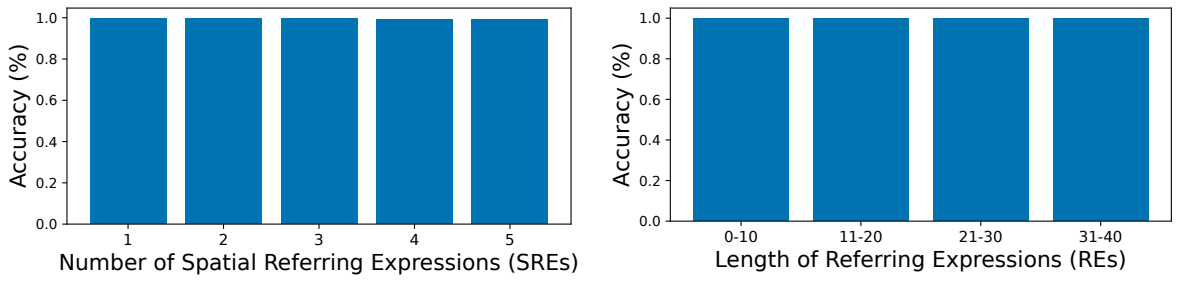
can use a proper modality to ground referring expressions to the correct landmarks.

To obtain semantically diverse spatiotemporal navigation commands, we first collected 1,723 spatial referring expressions (SREs) with respect to the robot's initial location from human users, then substituted the SREs in the 1,089 lifted natural language commands provided by Liu et al. [138]. The lifted commands cover 15 temporal patterns for common robotic tasks, each with 20 to 38 lifted commands. For example, given the lifted command "Visit a only then go to b", we can substitute the symbols a and b with the SREs "the vegan restaurant west of the bakery" and "the red car," respectively, to obtain the grounded natural language command "Visit the vegan restaurant west of the bakery only then go to the red car." We constructed 21,780 unique spatiotemporal language commands using five seeds to sample SREs for substitution. The commands contain varying numbers of SREs ranging from one to five.

4.5.2 Modular Evaluation

We first evaluated each module introduced in Section 4.4 on the semantically diverse dataset introduced in Section 4.5.1. All results were averaged over five seeds.

Spatial Referring Expression Recognition (SRER): We evaluated the LLM's ability to correctly extract all spatial referring expressions (SREs) from a natural language



(a) SRER Accuracy vs. Utterance Complexity (b) REG Accuracy vs. RE Complexity

Figure 4.5: Figure 4.5a shows the accuracies of the spatial referring expression recognition (SRER) module as the complexity of utterances (measured by the number of SREs in an utterance) increases. Figure 4.5b shows the top-10 accuracy of the referring expression grounding (REG) module as the complexity of REs (measured by string length) increases.

command and identify their spatial predicates described in Section 4.4.1, i.e., $\{r : (e_f, e_g)\}$ with spatial relation r, figure e_f and ground e_g. As shown in Table 4.1, the SRER module can reliably recognize SREs and their corresponding spatial predicates in language commands from unseen environments. Figure 4.5a further demonstrates that SRER achieves nearly perfect performance across commands with varying numbers of SREs. Occasionally, the SRER module fails to parse an SRE to the correct spatial predicate for an input command containing five long SREs.

Referring Expression Grounding (REG): We evaluated the REG module's ability to ground referring expressions, i.e., figures e_f's and grounds e_g's, to the correct physical landmarks described by text and images in the semantic map. We observe in Table 4.1 that the top-1 accuracy decreases as the number of landmarks increases from City 1 to City 4. Cities with more landmarks contain more instances that share similar textual or visual features. For example, there may be multiple cafe shops or red bicycles in a large environment. However, as we increase the number of top candidates from 1 to 10, REG achieves nearly perfect accuracy. Since the REG module provides candidate landmarks of figures and grounds to the SPG module (evaluated next), we hypothesize that as long as the correct landmark is among the top candidates, our system can still ground figures to the correct landmarks. We used 10 as the number of candidates for REG. Figure 4.5b shows that as the complexity of REs increases, the REG module consistently achieves near-perfect top-10 accuracies. These results align with those reported by Liu et al. [138].

Spatial Predicate Grounding (SPG): We evaluated whether the SPG module could identify the correct figure landmarks using spatial reasoning described in Section 4.4.3. As shown in Table 4.1, the SPG module performs uniformly well across all environments. The few failure cases were due to the instances where the distance between the figure and the ground landmarks was larger than the search threshold.

Lifted Translation (LT): Liu et al. [138] conducted a comprehensive evaluation of the generalization capability of various fine-tuned and pretrained LLMs for lifted translation.

We compared the accuracies of the best-performing model in [138], i.e., T5-base fine-tuned on a large composed dataset, and retrieval augmented generation (RAG) [160] with varying numbers of in-context examples. The fine-tuned model achieved the highest accuracies across all environments. As we increase the number of in-context examples for RAG from 10 to 100, the maximum tokens allowed by GPT-4 [104], we observe that the accuracy increases but is lower than that of the fine-tuned model. Thus, we used the fine-tuned T5-base model for lifted translation in our system. For cost effective reasons, we averaged the RAG results over two seeds per city.

4.5.3 Full System Evaluation and Ablation Study

We tested the overall performance of our language grounding system Lang2LTL-2, which takes a spatiotemporal navigation command as input and produces an LTL formula whose propositions are grounded to physical landmarks in the environment. The full system achieved an accuracy of $93.53 \pm 4.33\%$ on a dataset of 21,780 randomly sampled semantically diverse language commands.

To evaluate the effectiveness of using multimodal semantic information for language grounding, we conducted an ablation study where we only used one modality, i.e., text or images, in the referring expression grounding (REG) module. As shown in Figure 4.6, the full system using both modalities significantly outperformed the text-only and the image-only systems because either modality alone often did not provide enough semantic information. For example, an image of a restaurant front does not specify its cuisine being vegan, thus it will not be useful for grounding the referring expression "the vegan restaurant." However, the additional visual features provided by images can further disambiguate similar landmarks. For example, colors can help disambiguate a red and a yellow bicycle. In practice, detailed textual descriptions of landmarks are not always available, e.g., "the red brick building," but can be easily extracted from images by querying a pretrained VLM for image captions. Note that the text-only system is the

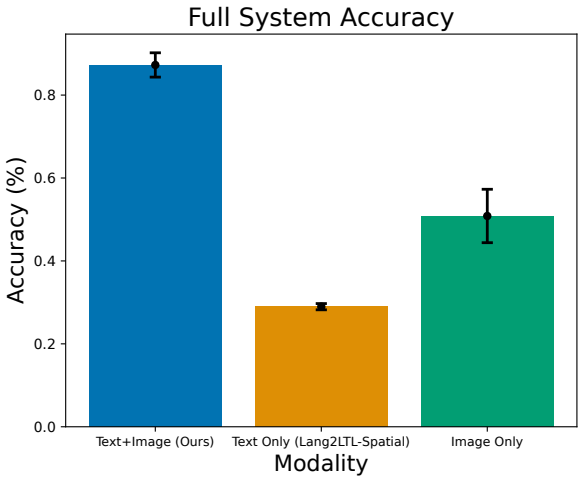


Figure 4.6: A comparison of the average accuracies of spatiotemporal language grounding systems using different modalities across four environments and five seeds per environment.

same as Lang2LTL [138] with an additional spatial reasoning capability. Liu et al. [138] showed that Lang2LTL outperforms Code-as-Policies [8], a prominent system that grounds natural language instructions to Python code directly executable on robots.

The accuracy of the spatial predicate grounding (SPG) module when given the top-10 candidate groundings from the referring expression grounding (REG) module was $97.26 \pm 2.07\%$. It supports our hypothesis that if the correct grounding landmark is among the top candidates output by the REG module, the SPG module can identify the correct figure landmark based on spatial reasoning.

4.6 Robot Demonstration

To demonstrate Lang2LTL-2's ability to inform an automated planner and enable the execution of spatiotemporal commands, we deploy the same system without modification at the task planning level on a quadruped robot Spot [132] in an indoor and outdoor environment. These environments contain nine and five objects, respectively, with multiple objects and landmarks that have similar textual or visual features, e.g., tables, couches, buildings, dumpsters, and cars.

We use Spot's GraphNav software to build a semantic map of the environment and capture images of landmarks and objects of interest. We only use the image modality for indoor experiments. For the outdoor environment, we additionally download textual descriptions of landmarks in the region from OpenStreetMap [161]. Given a grounded LTL task specification output by our language grounding system Lang2LTL-2, we use the AP-MDP planner [141] to produce a sequence of locations through the semantic map and Spot's onboard motion planner to move between two locations. We executed 50 semantically diverse spatiotemporal natural language commands on the physical robot. With the formal safety guarantee offered by LTL and the AP-MDP planner, the robot was able to abort the execution when a given task was infeasible. Please see the Additional Detail Section 4.9.4 for the list of all the language commands.

4.7 Related Work

Most existing robotic language grounding works focus on either spatial or temporal commands, with a few papers on grounding spatiotemporal commands to a limited capacity [15, 163].

4.7.1 Grounding Spatial Commands for Robots

SLOOP [135] is a system that grounds spatial commands in partially observable environments by using the spatial relations between a target object and multiple landmarks to construct an initial belief for a POMDP planner. LanguageRefer [164] is a learned transformer-based model that takes as inputs a spatial language command, a 3D point cloud of the scene, and bounding boxes of objects, then predicts the target object. RoboHop [165] builds a topological map of the environment with image segments as nodes. Like our work, RoboHop uses an LLM to extract referring expressions (REs) from a language command then a VLM to ground REs to nodes in the topological map.

4.7.2 Grounding Temporal Commands for Robots

Linear temporal logic (LTL) [88] is a mathematically precise language that can specify robotic tasks and provide satisfaction guarantees, especially for long-horizon, temporally-extended tasks with non-Markovian rewards. Early work of using LTL for temporal command grounding was limited to structured language [166]. Gopalan et al. [73] trained a Seq2Seq network [167] on natural language and LTL pairs in every new environment to ground language commands for navigation and manipulation. Like our work, Berg et al. [153] and Hsiung et al. [37] first translated commands to lifted LTL formulas then grounded the propositions to landmarks or objects but used a Seq2Seq network with limited capabilities.

To mitigate the lack of training data, Pan et al. [34] used an LLM to paraphrase structured language commands constructed from algorithmically generated LTL formulas. Patel et al. [36] and Wang et al. [35] proposed weakly supervised methods that use executed trajectories instead of LTL annotations to guide language grounding. Lang2LTL [138] is also a modular system that uses LLMs to ground temporally extended navigation commands in indoor and outdoor environments without retraining, given a text-based semantic database. However, Lang2LTL cannot ground spatial referring expressions or landmarks with visual descriptions. Our system Lang2LTL-2 improves upon Lang2LTL by incorporating spatial reasoning and using a VLM to process images.

4.7.3 Grounding Spatiotemporal Commands for Robots

Language commands from existing works of indoor [133, 134, 135, 168] and outdoor [136, 137] navigation are rich in spatial relations, but lack diverse temporal patterns. LM-Nav [137] uses an LLM to extract a sequence of referring expressions (REs) from a navigation command, then a VLM to ground the REs to images of physical landmarks. LM-Nav only grounds language commands of the sequenced visit type defined in [113]. VLMaps [169] fuses pretrained vision-language features with depth information to construct

a spatial map of the environment then directly indices a sequence of spatial referring expressions (SREs) extracted by an LLM in the map. LIMP [168] uses RGB-D information, an LLM, and a VLM to construct a 3D semantic map conditioned on the input language for motion planning to solve indoor mobile manipulation tasks. It translates free-form language commands to one of three temporal patterns using an LLM. Our system Lang2LTL-2 can ground language commands containing 15 temporal patterns commonly used in robotics [113]. An additional advantage of Lang2LTL-2 is its ability to ground REs that are not easily represented by visual description, e.g., generic referring expressions like "the vegan restaurant," and proper names like "Wildflour" (the name of a bakery) by using additional textual description from OpenStreetMap [161] in grounding city-scaled navigation commands.

4.8 Conclusion

We propose a modular language grounding system that consists of pretrained large language and vision-language models to ground spatiotemporal navigation commands to landmarks described by text and images in a semantic map of novel indoor and outdoor environments. We evaluate the individual modules and the full language grounding system on a semantically diverse dataset of 21,780 spatiotemporal navigation commands in four novel city-scaled environments. Our system achieved 93.53% accuracy, outperforming the previous SoTA. An autonomous robot with access to a semantic map and position tracking can use the same system without modification to follow spatiotemporal navigation commands in novel indoor and outdoor environments. We envision incorporating interaction with human users to further improve spatiotemporal language grounding.

4.9 Additional Details

4.9.1 Spatial Relations

We define the following spatial relations and rules for the spatial referring expression recognition (SRER) and the spatial predicate grounding (SPG) module. $\vec{v}_f$ and $\vec{v}_g$ represent the figure (from the ground landmark to the figure landmark) and the ground vector (from the ground landmark to the robot), respectively.

Table 4.2: Spatial Relations and Corresponding Rules.

Relation	*Rule*
left	$\vec{v}_f \in [\vec{v}_g - \pi, \vec{v}_g]$
right	$\vec{v}_f \in [\vec{v}_g, \vec{v}_g + \pi]$
in front of	$\vec{v}_f \in [\vec{v}_g - \pi/2, \vec{v}_g + \pi/2]$
opposite to	$\vec{v}_f \in [\vec{v}_g - \pi/2, \vec{v}_g + \pi/2]$
behind	$\vec{v}_f \in [\vec{v}_g - 3\pi/2, \vec{v}_g - \pi/2]$
near	$\vec{v}_f \in [\vec{v}_g - \pi, \vec{v}_g + \pi]$
next to	$\vec{v}_f \in [\vec{v}_g - \pi, \vec{v}_g + \pi]$
adjacent to	$\vec{v}_f \in [\vec{v}_g - \pi, \vec{v}_g + \pi]$
close to	$\vec{v}_f \in [\vec{v}_g - \pi, \vec{v}_g + \pi]$
by	$\vec{v}_f \in [\vec{v}_g - \pi, \vec{v}_g + \pi]$
north of	$\vec{v}_f \in [0, \pi]$
south of	$\vec{v}_f \in [\pi, 2\pi]$
east of	$\vec{v}_f \in [-\pi/2, \pi/2]$
west of	$\vec{v}_f \in [\pi/2, 3\pi/2]$
northeast of	$\vec{v}_f \in [0, \pi/2]$
northwest of	$\vec{v}_f \in [\pi/2, \pi]$
southeast of	$\vec{v}_f \in [\pi, 3\pi/2]$
southwest of	$\vec{v}_f \in [3\pi/2, 2\pi]$

The rule for the spatial relation between is that the figure landmark must lie between the two parallel lines perpendicular to the vector connecting and passing through the two grounding landmarks.

4.9.2 Details about Spatial Referring Expression Recognition (SRER)

The prompt for the spatial referring expression recognition (SRER) module is as follows,

Referring Expression Generation Prompt

```
Your task is to repeat exact strings from the given utterance which possibly refer
to certain propositions.

Utterance: move to red room
Propositions: red room

Utterance: visit Cutler Majestic Theater
Propositions: Cutler Majestic Theater

Utterance: robot c move to big red room and then move to green area
Propositions: big red room | green area

Utterance: you have to visit Panera Bread on Beacon Street, four or more than
four times Propositions: Panera Bread on Beacon Street

Utterance: go to Cutler Majestic Theater at Emerson College on Tremont Street,
exactly three times
Propositions: Cutler Majestic Theater at Emerson College on Tremont Street

...

Utterance: make sure you never visit St. James Church, a Christian place of
worship on Harrison Avenue, Dunkin' Donuts, Thai restaurant Montien, New Saigon
Sandwich, or Stuart St @ Tremont St
Propositions: St. James Church, a Christian place of worship on Harrison Avenue |
Dunkin' Donuts | Thai restaurant Montien | New Saigon Sandwich | Stuart St @ Tremont
```

```
St

Utterance:  move the robot through yellow region or small red room and then to
large green room
Propositions:  yellow region | small red room | large green room

Utterance:
```

4.9.3 Details about Lifted Translation (LT)

The prompt to GPT-4 for translating the lift command "navigate to a at most twice" using retrieval augmented generation (RAG) with 10 in-context examples is as follows,

Referring Expression Generation Prompt

```
Command:  navigate to a exactly twice
LTL formula:  M & a F & !  a F a | !  a G | a G | !  a G | a G ! a

Command:  navigate to a exactly three times
LTL formula:  M & a F & !  a F & a F & !  a F a | !  a G | a G | !  a G | a G | !  a
G | a G ! a

Command:  navigate to a exactly once
LTL formula:  M a | !  a G | a G ! a

Command:  visit a at most twice
LTL formula:  !  F & a U a & !  a U ! a F & a U a & !  a U ! a F a

Command:  go to a at most twice
LTL formula:  !  F & a U a & !  a U ! a F & a U a & !  a U ! a F a

Command:  navigate to a exactly five times
LTL formula:  M & a F & !  a F & a F & !  a F & a F & !  a F & a F & !  a F a | !
a G | a G | !  a G | a G | !  a G | a G | !  a G | a G | !  a G | a G ! a
```

```
Command:  navigate to a exactly four times
LTL formula:  M & a F & !  a F & a F & !  a F & a F & !  a F a | !  a G | a G | !  a
G | a G | !  a G | a G | !  a G | a G ! a

Command:  you can go to a at most twice
LTL formula:  !  F & a U a & !  a U ! a F & a U a & !  a U ! a F a

Command:  you can visit a at most twice
LTL formula:  !  F & a U a & !  a U ! a F & a U a & !  a U ! a F a

Command:  you can visit a at most twice
LTL formula:  !  F & a U a & !  a U ! a F & a U a & !  a U ! a F a

Command:
```

4.9.4 Language Commands for Robot Demonstration

Table 4.3 and 4.4 contain language commands and language grounding results for robot demonstrations in indoor and outdoor environments, respectively.

Table 4.3: Commands for Robot Demonstration in Indoor Environment

Navigational Command	Lang2LTL-2 Result
1.	success
2. go to the couch in front of the TV, the couch to the left of the kitchen counter, the kitchen counter between the couch and the refrigerator, the table next to the door, and the chair on the left of the bookshelf in any order	success
3. move to the couch in front of the TV, then the couch to the left of the kitchen counter, then the refrigerator right of the kitchen counter, then the table to the left of the bookshelf, then the table next to the door	success
4. walk to the chair in front of the bookshelf but only after the kitchen counter	success
5. move from the red couch near the TV to the blue couch left of the kitchen counter and only visit each landmark one time	success
6. go to the kitchen counter on the left side of the fridge exactly once and do not visit the kitchen counter on the left side of the fridge before the couch left of the TV	success
7. patrol the fridge right of kitchen counter at least five times	success
8. go to the kitchen counter between the blue couch and the fridge at most three times	success
9. visit the couch left of the counter, in addition never go to the TV in front of the couch	success
10. visit the blue couch near the kitchen counter then go to the fridge near the counter, in addition never visit the couch in front of the TV	success
11. visit the blue couch near the kitchen counter then go to the fridge near the counter, in addition never visit the couch in front of the TV or the kitchen counter	success
12. go to the refrigerator right of the kitchen counter, and only then go to the the couch left of counter, in addition always avoid the couch in front of the TV	success
13. go to the refrigerator near the kitchen counter then the couch left of the kitchen counter then the couch in front of the TV, in addition after going to the refrigerator near the kitchen counter, you must avoid the kitchen counter	success
14. go to the blue couch near the kitchen counter then the red couch in front of the TV, in addition go to the red couch in front of the TV only after visiting the refrigerator	success
15. go to the table near the door then the blue couch next to the kitchen counter then the red couch in front of the TV, in addition go to the table near the door only after visiting the bookshelf	success
16. go to the kitchen counter between the couch and fridge only after visiting the couch next to the counter, in addition go to the couch next to the counter only after visiting the refrigerator	success
17. go to the kitchen counter between the couch and the refrigerator exactly two times, in addition always avoid the TV in front of the couch	success
18. go to the bookshelf exactly two times, in addition always avoid the TV in front of the couch and the chair near the bookshelf	success
19. visit the couch next to the kitchen counter exactly two times, in addition do not go to the couch next to the kitchen counter before the couch near the TV	success
20. visit the couch in front of the TV at least two times, in addition do not go to the couch in front of the TV before the kitchen counter between the blue couch and the fridge	success
21. go to the refrigerator near the kitchen counter then the couch in front of the TV, in addition you must go to the couch left of the kitchen counter exactly twice	success
22. go to the blue couch next to the kitchen counter but never go to the blue couch next to the kitchen counter	unsatisfiable. abort correctly
23. go to the kitchen counter between the refrigerator and the blue couch at least twice, in addition go to the kitchen counter between the refrigerator and the blue couch exactly once	unsatisfiable. abort correctly
24. do not go to the desk near the door until the bookshelf, and do not go to the bookshelf until the desk near the door	unsatisfiable. abort correctly
25. find the kitchen counter between the refrigerator and the blue couch, in addition always avoid the red couch and the TV	unsatisfiable. abort correctly

Table 4.4: Commands for Robot Demonstration in Outdoor Environment

Navigational Command	Lang2LTL-2 Result
1. go to the stairs between the apartment building and the silver car	success
2. go to the stairs in front of the apartment building, the car to the right of the apartment and the car to the left of the apartment in any order	success
3. go to the white car to the right of the apartment, then visit the silver car near the stairs	success
4. move to the car east of the apartment, then the car west of the apartment, then dumpster north of the apartment, then red brick building north of the apartment	success
5. walk to the stairs between the apartment and the silver car, then the white car near the dumpster, then the dumpster	success
6. navigate to the dumpster north of the white car but do not visit the dumpster north of the white car before the stairs near the apartment	success
7. visit the the white car on the right side of the apartment but do not visit the white car on the right side of the apartment before the silver car on the east side of the apartment	success
8. only visit the apartment once, and do not visit the apartment until you visit the white car west of the apartment first	success
9. walk to the stairs in front of the apartment at least five times	success
10. visit the stairs between the apartment and the silver car at most three times	success
11. visit the silver car on the left side of the apartment exactly twice	success
12. go to the dumpster near the white car, in addition avoid the red brick building north of the apartment	success
13. go to the stairs between the silver car and the apartment then the apartment, in addition do not visit the dumpster north of the apartment	success
14. go to the white car left to the apartment and the silver car right to the apartment, in addition avoid the red brick building north of the apartment	success
15. move to the white car near the apartment at most twice, in addition avoid the dumpster near the white car	success
16. go to the white car near the dumpster exactly three times, in addition avoid stairs in front of the apartment	success
17. visit the dumpster near the white car, in addition you are not permitted to visit the apartment and the stairs between the apartment and silver car	success
18. navigate to the apartment at least two different occasions, in addition you are only permitted to visit the white car near the apartment at most once	success
19. walk to the white car close to the dumpster exactly once, in addition visit the stairs in front of the apartment on at least 2 separate occasions	success
20. visit the silver car east of the apartment exactly twice, in addition visit the red brick building in front of the apartment on at least 3 separate occasions	success
21. move to the stairs in front of the apartment then the apartment, in addition visit the white car near the dumpster exactly once	unsatisfiable. abort correctly
22. Visit the white car, then go to the red brick wall and then go to the silver car near the apartment, in addition you can never go to the apartment once you've seen the white car	unsatisfiable. abort correctly
23. go to the dumpster on the north side of the apartment, but always avoid the dumpster on the north side of the apartment	unsatisfiable. abort correctly
24. do not go to the white car west of the apartment until the red brick wall, and do not go to the red brick wall until the white car west of the apartment	unsatisfiable. abort correctly
25. find the dumpster near the white car, in addition avoid visiting the apartment and the red brick wall	unsatisfiable. abort correctly

CHAPTER 5

Skill Transfer Using Structured Task Representation

In Chapters 3 and 4, we answered the two questions of Robotic Language Grounding, namely, what grounding representation to use and how to ground language to the representation of choice. In this Chapter 5, we will show how to produce robot actions from the grounding representation linear temporal logic (LTL). LTL is a widely used task specification language with a compositional grammar that naturally induces commonalities among tasks while preserving safety guarantees. Deploying robots in real-world environments, such as households and manufacturing lines, requires generalization across novel task specifications without violating safety constraints. However, most prior work on reinforcement learning with LTL specifications treats every new task independently, thus requiring large amounts of training data to generalize. We propose LTL-Transfer, a zero-shot transfer algorithm that composes task-agnostic skills learned during training to safely satisfy a wide variety of novel LTL task specifications. Experiments in Minecraft-inspired domains show that after training on only 50 tasks, LTL-Transfer can solve over 90% of 100 challenging unseen tasks and 100% of 300 commonly used novel tasks without violating any safety constraints. We deployed LTL-Transfer at the task-planning

level of a quadruped mobile manipulator to demonstrate its zero-shot transfer ability for fetch-and-deliver and navigation tasks.

5.1 Introduction

Deploying robots in the real world requires generalization across many novel tasks while preserving safety. For example, an industrial robot that fetches the same components in different orders based on the assembled product should only learn to fetch each part once. These tasks typically share constituents like objects and trajectory segments, which creates an opportunity to reuse knowledge [170].

Linear temporal logic (LTL) [88] is an effective means of specifying objectives, including safety constraints, for reinforcement learning (RL) agents [140, 171, 172]. Its compositional grammar reflects the compositional nature of most tasks. However, most prior approaches to RL for LTL tasks learn to solve every new task from scratch. We propose a zero-shot transfer algorithm, LTL-Transfer, that exploits the compositionality of LTL task specification to safely solve novel tasks without additional training by composing skills learned in prior tasks. Transferring skills is more data-efficient than learning from scratch and more computationally efficient than planning.

(a) go to shelf to fetch book (b) deliver juice to desk

Figure 5.1: The robot is executing four task-agnostic skills sequentially to solve a novel task $\mathbf{F}(book \wedge \mathbf{F}(desk_a \wedge \mathbf{F}(juice \wedge \mathbf{F}desk_a)))$, i.e., fetch and deliver a book then a juice bottle to the user. Two of the four skills are shown.

We show in a simulated Minecraft-inspired domain that LTL-Transfer can solve over 90% of 100 challenging unseen tasks and 100% of 300 common and less complex

novel tasks after training on only 50 tasks and never violates a safety constraint. We deploy LTL-Transfer at the task-planning level of a quadruped mobile manipulator to solve fetch-and-deliver and navigation tasks in zero-shot. Our key insight is efficiently reusing learned skills by leveraging similarities between the novel and training tasks. Code, datasets, supplementary materials, and robot demonstration videos are at `https://jasonxyliu.github.io/LTL-Transfer`.

5.2 Preliminaries

Linear Temporal Logic (LTL) for Task Specification: LTL is a widely used alternative to numerical rewards for task specification. An LTL formula φ is a Boolean function that determines whether a given trajectory satisfies the objective expressed by the formula. Littman et al. [171] showed that LTL can express non-Markovian, temporal tasks that numerical rewards cannot, and it has become a target language for acquiring task specification in many settings, including from natural language [20, 38, 73, 153] and learning from demonstration [173, 174, 175]. Formally, an LTL formula is interpreted over traces of Boolean propositions over discrete time and is defined through the following recursive syntax:

$$\varphi := \alpha \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \mathbf{X}\varphi \mid \varphi_1 \ \mathbf{U} \ \varphi_2. \tag{5.1}$$

$\alpha \in \boldsymbol{\alpha}$ is a proposition that maps a state to a Boolean value; φ, φ_1, and φ_2 are valid LTL formulas. The operator $\mathbf{X}$ (next) is used to define a property $\mathbf{X}\varphi$ that holds if φ holds at the next time step. The formula $\varphi_1 \ \mathbf{U} \ \varphi_2$ with the binary operator $\mathbf{U}$ (until) specifies that φ_1 must hold until φ_2 first holds at a future time. The operators $\neg$ (not) and $\vee$ (or) are identical to propositional logic operators. We also utilize the following abbreviated operators: $\wedge$ (and), $\mathbf{F}$ (finally or eventually), and $\mathbf{G}$ (globally or always). $\mathbf{F}\varphi$ specifies that the formula φ must hold at least once in the future, and $\mathbf{G}\varphi$ specifies that φ must always hold in the future. Consider the Minecraft map depicted in Figure 5.2.

The task of collecting *wood* and *axe* in an arbitrary order is specified by the LTL formula $\mathbf{F}axe \wedge \mathbf{F}wood$. The formula $\mathbf{F}(axe \wedge \mathbf{F}wood)$ specifies collecting *wood* at least once after collecting *axe*. $\mathbf{F}wood \wedge \neg wood\ \mathbf{U}\ axe$ specifies the task of collecting *wood* only after *axe* has been collected.

Every LTL formula can be represented as a Büchi automaton [114, 176] interpreted over an infinite trace of truth values of the propositions in the formula, thus providing an automated translation of an LTL specification to a transition-based representation. We consider task specification in the co-safe fragment of LTL [177, 178] where formulas can be verified by a finite trace, thus making it ideal for episodic tasks. Camacho et al. [140] showed that every co-safe LTL formula φ can be translated to an equivalent reward machine (RM) [142, 155], $\mathcal{M}_\varphi = \langle \mathcal{Q}_\varphi, q_{0,\varphi}, \mathcal{Q}_{term,\varphi}, \varphi, T_\varphi, R_\varphi \rangle$; where $\mathcal{Q}_\varphi$ is the finite set of states, $q_{0,\varphi}$ is the initial state, $\mathcal{Q}_{term,\varphi}$ is the set of terminal states; $T_\varphi : \mathcal{Q}_\varphi \times 2^{|\boldsymbol{\alpha}|} \to \mathcal{Q}_\varphi$ is the deterministic transition function; and $R_\varphi : \mathcal{Q}_\varphi \to \mathbb{R}$ represents the reward accumulated by entering a given state. Figure 5.2d shows the reward machine graph representing the LTL formula $\mathbf{F}wood \wedge \neg wood\ \mathbf{U}axe$. Nodes encode progressed RM states, 0 for an accepting state and 3 for a failure state. Boolean formulas label edges. Truth assignments of propositions $\boldsymbol{\alpha}$ that satisfy edges induce transitions in the RM. Our proposed algorithm, LTL-Transfer, for transferring learned policies to solve novel LTL specifications, is compatible with all algorithms that generate policies by solving a product MDP of the reward machine $\mathcal{M}_\varphi$ and the task environment.

Options Framework: Sutton and Barto [179] introduced a framework, termed options or skills, for incorporating temporally extended actions into reinforcement learning. An option $o = \langle \mathcal{I}, \beta, \pi \rangle$ is defined by the initiation set $\mathcal{I}$, a set of states where the option policy can be executed; the termination condition β, which determines when the execution ends; and the option policy π. Our proposed algorithm, LTL-Transfer, compiles policies learned during training into task-agnostic options that are transferred to solve novel tasks.

5.3 Related work

Most approaches extending reinforcement learning (RL) to temporal tasks first generate a product MDP of the state space and the automaton equivalent of the LTL task specification [140, 142, 155, 171, 172]. Notably, Jothimurugan et al. [180] proposed interleaving graph-based planning on the automaton with RL to bias exploration towards trajectories that satisfy the LTL specification. Although these approaches exploited the compositional structure of LTL to accelerate learning, they did not exploit the compositionality to transfer to novel tasks. Thus, the policy to satisfy a novel LTL specification must be learned from scratch.

A common approach towards generalization across temporal tasks has been to learn independent policies for subtasks [181, 182, 183, 184]. Given a new task, these methods then sequentially compose the learned policies in an admissible order. Consider the Minecraft-inspired grid world depicted in Figure 5.2a containing *wood* and *axe* objects. The subtask-based methods learn policies to solve subtasks $\mathbf{F}\mathit{wood}$ and $\mathbf{F}\mathit{axe}$ involving reaching each object. When tasked with the specification $\varphi_{test} = \mathbf{F}\mathit{wood} \wedge (\neg \mathit{wood}\, \mathbf{U}\, \mathit{axe})$, i.e., collect *wood*, but do not collect *wood* until *axe* is collected, the subtask-based methods would violate the ordering constraint by reaching *axe* through the grid cells containing *wood*. These approaches rely on additional fine-tuning to solve the target task correctly. We propose a general framework for transferring learned policies to novel tasks in zero-shot without violating any safety constraints.

Kuo et al. [185] proposed learning subnetworks for propositions and operators, then creating the final policy network through composition. Vaezipoor et al. [186] proposed learning a latent embedding over LTL formulas using a graph neural network to solve novel LTL tasks. In contrast, our method uses formal methods to identify learned policies best suited for transfer, thus requiring orders of magnitude fewer training tasks to achieve comparable results. Furthermore, neither method can guarantee the preservation of safety constraints like our approach. Xu et al. [187] considered transfer learning between pairs of

source and target tasks, while our approach trains on a collection of tasks and transfers to a set of novel tasks. Nangue Tasse et al. [188] attempted zero-shot composition through logical and temporal compositions of policies, but their approach assumes that a given task must be satisfiable. By leveraging the structure of the task automaton, our proposed algorithm aborts execution immediately after it identifies the task as unsolvable given the learned skills.

A qualitatively distinct approach to zero-shot generalize to novel tasks is model-based RL that plans with an estimated transition model [189, 190]. Compared to model-based methods, our approach transfers learned policies in zero-shot thus requires significantly less computation at test time.

Our approach draws inspiration from prior works on learning portable skills in Markov domains [191, 192, 193, 194]. These approaches learn task-agnostic representations of preconditions, constraints, and effects of an option [179]. We learn portable skills to satisfy novel LTL task specifications.

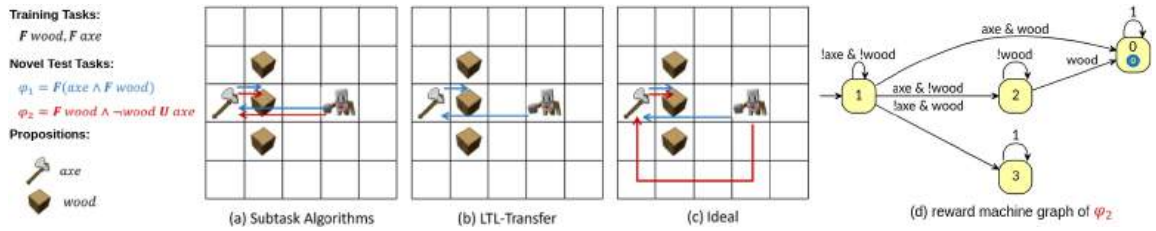


Figure 5.2: An example of tasks, the environment, and trajectories. The robot learned to solve the two training LTL tasks and is expected to solve two novel tasks φ_1 and φ_2. Figure 2a depicts the trajectories output by a subtask-based algorithm (blue for φ_1, red for φ_2). Figure 2b depicts the trajectories produced by our proposed algorithm LTL-Transfer. Note that LTL-Transfer does not start execution for the task φ_2 as the two learned policies do not guarantee the preservation of the ordering constraint. Figure 2c depicts the optimal trajectories for the novel tasks φ_1 and φ_2. Figure 2d is a graph representation of the reward machine (RM) for the task φ_2. Nodes represent RM states. Edges represent Boolean formulas.

5.4 Problem definition

We represent the environment as an MDP without the reward function $\mathcal{M}_{\mathcal{S}} = \langle \mathcal{S}, \mathcal{A}, T_{\mathcal{S}} \rangle$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, and $T_{\mathcal{S}} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ represents the transition dynamics of the environment which we assume to be hidden from the RL algorithm. Further, a set $\boldsymbol{\alpha}$ of Boolean propositions represents the facts about the environment and forms the compositional building blocks for specifying tasks. We assume a labeling function $L : \mathcal{S} \to 2^{|\boldsymbol{\alpha}|}$ that maps the state to the Boolean propositions is given. A task in the environment $\mathcal{M}_{\mathcal{S}}$ is specified by a linear temporal logic (LTL) formula φ, and it is translated to a reward machine $\mathcal{M}_{\varphi} = \langle \mathcal{Q}_{\varphi}, q_{0,\varphi}, \mathcal{Q}_{term,\varphi}, \varphi, T_{\varphi}, R_{\varphi} \rangle$ detailed in Section 5.2.

Given a set of training tasks $\Phi_{train} = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$, specified by LTL, and policies for a set of options $\mathcal{O}_q$ learned from these training tasks, a zero-shot transfer algorithm needs to solve a novel LTL task $\varphi_{test} \notin \Phi_{train}$ in the same environment without additional training.

5.5 LTL-Transfer with transition-centric options

5.5.1 Algorithm Overview

Consider the environment map depicted in Figure 5.2b. Assume an RL algorithm has learned option policies to collect *axe* and *wood* individually from the training tasks $\mathbf{F}axe$ and $\mathbf{F}wood$, respectively. Now given the test task $\varphi_1 = \mathbf{F}(axe \wedge \mathbf{F}\ wood)$, i.e., first collect *axe*, then *wood*, a transfer algorithm should identify that sequentially composing the policies for $\mathbf{F}axe$ and $\mathbf{F}wood$ solves the new task φ_1 (as depicted in blue). Then consider a different test task $\varphi_2 = \mathbf{F}wood \wedge \neg wood\ \mathbf{U} axe$, i.e., first collect *axe*, then *wood*, but avoid *wood* until *axe* is collected. The transfer algorithm must realize that the policy that satisfies $\mathbf{F}axe$ does not guarantee avoiding *wood* while going to *axe*. Therefore, it must not start execution using only these two learned policies to avoid accidentally violating

the ordering constraint.

We developed a zero-shot transfer algorithm, LTL-Transfer, that composes learned policies to solve novel LTL tasks while enforcing ordering constraints. It operates in two stages.

1. First, LTL-Transfer accepts the set of training tasks Φ_{train} and the policies learned from the training tasks and compiles a set of task-agnostic, portable options $\mathcal{O}_e$.
2. Next, it identifies and executes a sequence of options from the set $\mathcal{O}_e$ to solve a novel task φ_{test}.

We can use a class of RL algorithms that operate on a product MDP composed of the environment $\mathcal{M}_{\mathcal{S}}$ and the reward machine $\mathcal{M}_{\varphi}$ to learn option policies from the training LTL tasks [140, 142, 155, 171, 172]. We chose LPOPL [172] because it explicitly allows for sharing policies across LTL task specifications that share progression states. Given a set of LTL tasks, LPOPL first translates each task to a reward machine (RM) and decomposes tasks to subtasks, each of which corresponds to a state in the RM, then learns an action-value function, represented by a DQN [195], for each subtask. Please see the supplementary materials for implementation details of LPOPL. The learned policy is Markov with respect to the environment states $\mathcal{S}$ for a given RM state, i.e., the policy to be executed in the RM state $q \in \mathcal{Q}_{\varphi}$ is $\pi^{\varphi}_{q} : \mathcal{S} \to \mathcal{A}$.

Executing the state-centric option $o^{\varphi}_{q} \in \mathcal{O}_q$ with the policy π^{φ}_{q} from the reward machine state q triggers a transition in the RM on a path to an acceptance state. There can be multiple outgoing transitions from an RM state, so the target RM transition of an option o^{φ}_{q} is conditioned on the environment state $s \in \mathcal{S}$ where the execution was initiated. We propose compiling each state-centric option, o^{φ}_{q}, into multiple transition-centric options by partitioning the initiation set of the state-centric option based on the estimated success probability of its policy satisfying the target RM transition from the starting environment state. Each resulting transition-centric option will maintain the satisfaction

of self-transition edge $e^{\varphi}_{q,q}$ from the starting RM state q until it triggers the target RM transition $e^{\varphi}_{q,q'}$. Our insight is that each transition-centric option triggers a transition in the reward machine on a path to an acceptance state, and these RM transitions may be shared across different tasks. Thus, the transition-centric options $\mathcal{O}_e$ are portable across different tasks. We describe the compilation algorithm in Section 5.5.2.

Given a novel LTL task specification $\varphi_{test} \notin \Phi_{train}$, our transfer algorithm first constructs a reward machine representing the task, $\mathcal{M}_{\varphi_{test}}$, then identifies a path through the reward machine that can be traversed by sequentially executing transition-centric options from the set $\mathcal{O}_e$. Our transfer algorithm is sound, and it terminates. We describe the details of the transfer algorithm in Section 5.5.3.

The key advantage of our approach is that the option compilation can be done offline for any environment. We can then transfer the options to solve novel tasks at execution time in zero-shot. Thus, learning to solve a limited number of LTL tasks can help solve a wide gamut of unseen tasks.

5.5.2 Compilation of Transition-Centric Options

The policy learned to satisfy an LTL task specification φ identifies the current reward machine state $q \in \mathcal{Q}_\varphi$ and executes a Markov policy π^{φ}_{q} until the state of the reward machine progresses. We represent this policy as a state-centric option, $o^{\varphi}_{q} = \langle \mathcal{S}, \beta_{e^{\varphi}_{q,q}}, \pi^{\varphi}_{q} \rangle$, where the initiation set is the entire state space $\mathcal{S}$ of the environment; the option terminates when the truth assignment of the propositions $\boldsymbol{\alpha}$ violates the self-transition $e^{\varphi}_{q,q}$ from the RM state q. The termination condition $\beta_{e^{\varphi}_{q,q}}$ is formally defined as follows,

$$\beta_{e^{\varphi}_{q,q}} = \begin{cases} 1, & \text{if } L(s) \not\models e^{\varphi}_{q,q} \\ 0, & \text{otherwise.} \end{cases} \tag{5.2}$$

A transition-centric option $o_{e^{\varphi}_{q,q}, e^{\varphi}_{q,q'}}$ executes a Markov policy that ensures that the

Algorithm 1 Compile State-Centric Options to Transition-Centric Options

function COMPILE($\mathcal{M}_{\mathcal{S}}, \Phi_{train}, \mathcal{O}_q$)
  $\mathcal{O}_e \leftarrow \emptyset$
  for $\varphi \in \Phi_{train}$ **do**
    $\mathcal{M}_\varphi \leftarrow$ GENERATERM(φ)
    $\mathcal{O}_q^\varphi \leftarrow \left\{ o_q^{\varphi'} \in \mathcal{O}_q : \varphi' = \varphi \right\}$
    for $o_q^\varphi = \langle \mathcal{S}, \beta_{e_{q,q}^\varphi}, \pi_q^\varphi \rangle \in \mathcal{O}_q^\varphi$ **do**
      $\mathcal{Q}_{out} = \{q' : q' \text{ is an out-neighbor of } q\}$
      $\forall q' \in \mathcal{Q}_{out} : \mathcal{E} \leftarrow \left\{ (e_{q,q}^\varphi, e_{q,q'}^\varphi) : e_{q,q}^\varphi \text{ is the self edge} \right\}$
      for $s \in \mathcal{S}$ **do**
        Generate N_r rollouts from s using π_q^φ
        Record edge transition frequencies $n_s(e_{q,q'}^\varphi)\ \forall\ (e_{q,q}^\varphi, e_{q,q'}^\varphi) \in \mathcal{E}$
        $\forall\ q' \in \mathcal{Q}_{out} : f_{e_{q,q'}^\varphi}(s) \leftarrow \frac{n_s(e_{q,q'}^\varphi)}{N_r}$
      $\mathcal{O}_{q,e}^\varphi \leftarrow \left\{ o_{e_{q,q}^\varphi, e_{q,q'}^\varphi} = \langle \mathcal{S}, \beta_{e_{q,q}^\varphi}, \pi_q^\varphi, e_{q,q}^\varphi, e_{q,q'}^\varphi, fe_{q,q'}^\varphi \rangle \right\}$
      $\mathcal{O}_e \leftarrow \mathcal{O}_e \cup \mathcal{O}_{q,e}^\varphi$
  return $\mathcal{O}_e$

truth assignment of the propositions $\boldsymbol{\alpha}$ satisfies the self-transition edge $e_{q,q}^\varphi$ at all time until the policy yields a truth assignment that satisfies the target outgoing edge $e_{q,q'}^\varphi$. We define a transition-centric option as follows,

$$o_{e_{q,q}^\varphi, e_{q,q'}^\varphi} = \langle \mathcal{S}, \beta_{e_{q,q}^\varphi}, \pi_q^\varphi, e_{q,q}^\varphi, e_{q,q'}^\varphi, f_{e_{q,q'}^\varphi} \rangle. \tag{5.3}$$

The initiation set is the entire state space $\mathcal{S}$ of the environment; the termination condition $\beta_{e_{q,q}^\varphi}$ is defined by the violation of the self-transition edge $e_{q,q}^\varphi$; the option's policy is Markov $\pi_q^\varphi : \mathcal{S} \to \mathcal{A}$; $e_{q,q}^\varphi$ and $e_{q,q'}^\varphi$ represent the self-transition and the target outgoing edge, respectively; and $f_{e_{q,q'}^\varphi} : \mathcal{S} \to [0, 1]$ represents the success probability of the policy π_q^φ satisfying the target edge $e_{q,q'}^\varphi$ starting from the environment state $s \in \mathcal{S}$.

Algorithm 1 describes our approach to compiling each state-centric option o_q^φ to a set of transition-options $\left\{ o_{e_{q,q}^\varphi, e_{q,q'}^\varphi} : q \in \mathcal{Q}_\varphi, q' \text{is out-neighbor of } q \right\}$. Executing the policy π_q^φ of the state-centric option may satisfy different outgoing edges $e_{q,q'}^\varphi$ of the reward machine state q depending on what environment state $s \in \mathcal{S}$ the execution was initiated. Thus, the success probability $f_{e_{q,q'}^\varphi}$ acts as a soft segmenter of the state space $\mathcal{S}$; it can be estimated by policy rollouts from all environment states in discrete domains or using sampling-based methods [193, 194] in continuous domains.

Algorithm 2 Zero-shot transfer to test task φ_{test}

1: **function** TRANSFER($\mathcal{M}_{\mathcal{S}}$, φ_{test}, $\mathcal{O}_e$)
2: $\quad\mathcal{M}_{\varphi_{test}} \leftarrow$ GENERATE_RM(φ_{test})
3: $\quad\mathcal{M}_{\varphi_{test}} \leftarrow$ PRUNE($\mathcal{M}_{\varphi_{test}}$)
4: $\quad s \leftarrow$ INITIALIZE($\mathcal{M}_{\mathcal{S}}$)
5: $\quad q \leftarrow q_{0,\varphi_{test}}$
6: $\quad$**while** $q \neq q^{\top}$ **do**
7: $\quad\quad P_{\text{cache}} \leftarrow \{p_i : p_i = [e_0, \ldots, e_{n_i}]$ path from q to $q^{\top}$ in $\mathcal{M}_{\varphi_{test}}\}$
8: $\quad\quad \mathcal{O}_{p[0]} = \{o_{e_1,e_2} \in \mathcal{O}_e : \text{MATCHEDGE}((e_1,e_2),(e^{\varphi_{test}}_{q,q}, p[0]))\}\ \forall p \in P_{\text{cache}}$
9: $\quad\quad \mathcal{O}_{[0]} = \bigcup_p \mathcal{O}_{p[0]}$
10: $\quad\quad \langle s', q' \rangle \leftarrow \langle s, q \rangle$
11: $\quad\quad$**while** $\mathcal{O}_{[0]} \neq \emptyset$ and $q' = q$ **do**
12: $\quad\quad\quad o^*_{e_1,e_2} \leftarrow \arg\max_{o_{e_1,e_2} \in \mathcal{O}_{[0]}} f_{e_2}(s)$
13: $\quad\quad\quad \langle s', q' \rangle \leftarrow$ EXECUTE(π^*)
14: $\quad\quad\quad$**if** $q' = q$ **then**
15: $\quad\quad\quad\quad \mathcal{O}_{[0]} \leftarrow \mathcal{O}_{[0]} \setminus o^*_{e_1,e_2}$
16: $\quad\quad$**if** $q' = q$ **then**
17: $\quad\quad\quad$**return** Failure
18: $\quad\quad$**else**
19: $\quad\quad\quad \langle s, q \rangle \leftarrow \langle s', q' \rangle$
20: $\quad$**return** Success

5.5.3 Transferring to Novel LTL Task Specification

Algorithm 2 describes the zero-shot transfer algorithm that composes transition-centric options from the set $\mathcal{O}_e$ to solve a novel test task φ_{test} in the environment $\mathcal{M}_{\mathcal{S}}$. Line 2 generates the reward machine (RM) graph for the test task. Line 3 examines each edge $e^{\varphi_{test}}_{q,q'}$ of the RM, identifies the transition-centric options that satisfy that edge transition, and prunes an edge if no such option is found. Line 7 identifies and caches all paths from the current state q to the accepting state $q^{\top}$ of the reward machine. Lines 8 and 9 identify a set of all eligible options that can potentially achieve an RM transition from the current state to a progressed state on a path to the accepting state.

Lines 12 and 13 then execute the option with the highest success probability of satisfying the target edge transition, estimated by f. If the option fails to progress to another RM state, we delete it from the set (Line 15) and execute the next option. If the set of eligible options is empty at any point without reaching the accepting RM state $q^{\top}$, Line 17 exits with a failure. A successful transfer occurs when the RM progresses to the accepting state $q^{\top}$.

5.5.4 Matching Options to Reward Machine Transitions

The edge-matching conditions determine whether we can safely apply a transition-centric option to transition along an edge of the reward machine (RM). We used the edge-matching conditions to prune the RM graph to retain only the edges matched with available options (Algorithm 2 Line 3) and identify eligible options from a given RM state (Algorithm 2 Line 8). Given a test task φ_{test}, when the current RM state is q, its self-transition edge is $e_{q,q}^{\varphi_{test}}$ and the target outgoing edge is $e_{q,q'}^{\varphi_{test}}$, to determine if a transition-centric option o_{e_1,e_2} matches the target transition in the RM, we propose two edge-matching conditions, *Constrained* and *Relaxed*. Both ensure the task execution does not fail due to an unsafe transition.

Constrained Edge-Matching Condition requires that every truth assignment satisfying the self-transition edge e_1 of the option also satisfies the self-transition edge $e_{q,q}^{\varphi_{test}}$ of the reward machine $\mathcal{M}_{\varphi_{test}}$. Similarly, every truth assignment satisfying the outgoing edge e_2 of the option must satisfy the target transition $e_{q,q'}^{\varphi_{test}}$ of the test task's RM. This strict requirement reduces the applicability of the learned options but ensures that the target edge is always achieved. The *Constrained* edge-matching condition is satisfied if the following Boolean expression holds:

$$\neg sat(e_1 \wedge \neg e_{q,q}^{\varphi_{test}}) \ \wedge \ \neg sat(e_2 \wedge \neg e_{q,q'}^{\varphi_{test}}). \tag{5.4}$$

Let $sat(g)$ be a Boolean function that is true if and only if a truth assignment exists to satisfy the Boolean formula g.

Relaxed Edge-Matching Condition requires the self edges e_1 and $e_{q,q}^{\varphi_{test}}$ share satisfying truth assignments, so must the outgoing edges e_2 and $e_{q,q'}^{\varphi_{test}}$. However, it allows the option to have valid truth assignments that may not satisfy the target outgoing edge, yet none of the truth assignments should trigger a transition to an unrecoverable failure state $q^{\perp}$. We identify $q^{\perp}$ as a sink state of the RM graph without outgoing edges. Further,

all truth assignments that terminate the option must not satisfy the self-transition edge of the test task's reward machine. The *Relaxed* edge-matching condition can retrieve more eligible options. It is satisfied if the following Boolean expression holds:

$$sat(e_1 \wedge e_{q,q}^{\varphi_{test}}) \ \wedge \ sat(e_2 \wedge e_{q,q'}^{\varphi_{test}}) \wedge \\ \neg sat(e_1 \wedge e^{\perp}) \ \wedge \ \neg sat(e_2 \wedge e^{\perp}) \ \wedge \ \neg sat(e_2 \wedge e_{q,q}^{\varphi_{test}}). \quad (5.5)$$

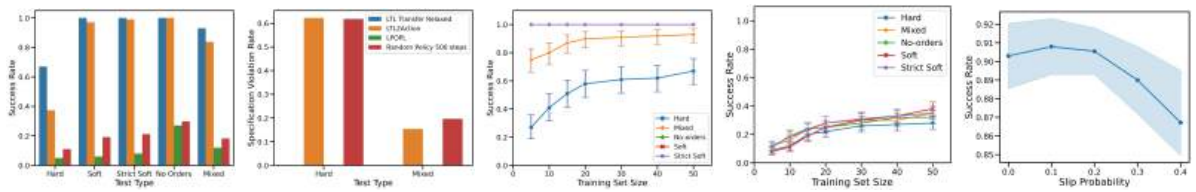
5.5.5 Optimization

We parallelized the two key computational bottlenecks of LTL-Transfer, i.e., the estimation of the success probability $f_{e_{q,q'}^{\varphi}}$ and the evaluation of the edge-matching condition since there are no shared memory requirements. We implemented the *Relaxed* edge-matching condition using a propositional model counting algorithm [196] from SymPy [197], and the *Constrained* edge-matching condition using string comparison to circumvent the model counting problem and found a significant speed-up.

5.6 Experiments

The aim of our evaluation is to test the hypothesis that LTL-Transfer can efficiently transfer learned skills to solve novel temporal tasks while preserving safety guarantees. We tested our hypothesis in simulation and on a physical robot. The simulation domain allows us to scale skill transfer across a wide variety of tasks, while the real robot demonstrates the practicality of our approach for real-world mobile manipulation tasks. We first defined five types of specifications of varying complexity for training and testing, then conducted experiments to evaluate the following hypotheses,

1. **H1:** LTL-Transfer exceeds the baselines' success rates of solving novel tasks.
2. **H2:** *Relaxed* edge-matching condition leads to higher success rates than the *Constrained* condition.



(a) Success Rate (b) Violation Rate (c) *Relaxed* Match (d) *Constrained* Match (e) Stochastic Domain

Figure 5.3: Figure 5.3a shows the success rates of four methods on five test sets after training on the *Mixed* training set. Figure 5.3b depicts their specification violation rates (with a shared legend). Figure 5.3c and 5.3d show the success rates of LTL-Transfer on five test sets after training on *Mixed* sets of various sizes using *Relaxed* and *Constrained* edge-matching condition, respectively. The error bars depict the 95% credible interval if the successful transfer was modeled as a Bernoulli distribution. Figure 5.3e shows the success rate as the slip probability increases averaged over four maps.

3. **H3:** Transferring learned policies to certain specification types (introduced in Section 5.6.2) leads to higher success rates.

4. **H4:** LTL-Transfer is robust to highly uncertain transition dynamics.

5.6.1 Task Environment

We tested LTL-Transfer in a Minecraft-inspired grid-world domain commonly seen in compositional reinforcement learning (RL) and integration of temporal logics with RL [172, 180, 183, 184]. This domain is particularly well suited for testing transfer learning with temporal tasks as policies to solve individual tasks can be learned rapidly with few computational resources. Thus, we can run comprehensive evaluations of skill transfer across a full factorial of training and test task types that cover a wide variety of LTL templates. We conducted experiments on four maps, similar to that depicted in Figure 5.2, with a dimension of 19×19. Each location in the map is either vacant or occupied by one of nine object types. Multiple instances of an object type may occur across the map. After the agent enters a grid cell occupied by an object, the proposition representing that object type becomes true. Actions are moving in four cardinal directions; an invalid action results in no movement. We tested zero-shot transfer in both deterministic and stochastic domains.

5.6.2 Types of Task Specifications

For a comprehensive evaluation of transferring learned policies across various LTL specifications, we considered the following three types of ordering constraints, proposed by Shah et al. [173], which constitute five specification types. Each constraint is defined on a pair of propositions a and b. Without loss of generality, we assume that a precedes b.

1. **Hard** ordering constraints occur when b must never be true before a. This property is expressed through the LTL formula $\neg b \ \mathbf{U} \ a$.

2. **Soft** ordering constraints allow b to occur before a as long as b happens at least once after a becomes true for the first time. Soft orders are expressed through the LTL formula $\mathbf{F}(a \ \wedge \ \mathbf{F}b)$.

3. **Strictly Soft** ordering constraints are similar to soft orders except that b must be true strictly after a first holds. Thus, a and b holding simultaneously would not satisfy a strictly soft order. Strictly soft orders are expressed through the LTL formula $\mathbf{F}(a \ \wedge \ \mathbf{XF}b)$.

We sampled five training sets, with 50 formulas in each, that represent five different specification types, *Hard*, *Soft*, *Strictly-Soft*, *No-Orders*, and *Mixed*, and a test set of 100 formulas for each type. This imitates the real-world scenario where users do not know the test task beforehand, so the robot must be trained on a limited set of training tasks then transfer to a wide variety of novel tasks. When constructing a test set, we excluded tasks already in the training set. All specifications in the *Hard*, *Soft*, and *Strictly-Soft* sets were expressed using the respective templates described above. *No-Orders* sets have no ordering constraints, e.g., $\mathbf{F}a \ \wedge \ \mathbf{F}b \ \wedge \ \mathbf{F}c$. In the *Mixed* set, each binary precedence constraint was expressed as one of the three ordering constraints described above. A given task in the environment involves visiting a specified set of object types in an admissible order determined by ordering constraints. Please see the supplementary materials for example formulas from each specification type.

5.6.3 Results and Discussion

Comparison with Baselines: We compare the performance of LTL-Transfer to three baselines, i.e., LTL2Action [186], LPOPL [172], and a random policy.

LTL2Action proposed by Vaezipoor et al. [186] embeds LTL specifications using a graph neural network and sequentially selects the next proposition to satisfy. This pre-trained embedding is appended to the state features to yield a goal-conditioned task policy, termed LTL Bootcamp, which serves as the upper bound of LTL2Action's performance on novel tasks. We trained the LTL Bootcamp on the same training sets as LTL-Transfer and compared their performance.

LPOPL, detailed in Section 5.5.1, serves as the lower bound that any transfer algorithm must surpass due to its limited transfer ability to solve novel tasks. While LPOPL was not explicitly designed for zero-shot transfer, it can satisfy task specifications that are a progression of a training LTL formula because of its use of LTL progression and multi-task learning.

Figure 5.3a depicts the success rate of all the methods on 100 novel tasks in each test set described in Section 5.6.2 after training on 50 tasks of the *Mixed* type. LPOPL performed worse than the random policy as it did not attempt to satisfy any specification that did not exist in its progression set. Both LTL-Transfer and LTL2Action have near-perfect success rates on *Soft*, *Strictly-Soft*, and *No-Orders* test set as these specifications do not have irrecoverable failure state. Specifications in *Hard* and *Mixed* test sets have failure states. Thus, we observe a lower success rate across all methods. However, LTL-Transfer demonstrates the best transfer success rate in the difficult test sets. Crucially, Figure 5.3b shows that LTL-Transfer never violated any specification, while LTL2Action's specification violation rate is similar to that of the random policy, which could mean that LTL2Action essentially acts randomly given an infeasible task.

Effect of Edge-Matching Condition: We trained LTL-Transfer on *Mixed* training

sets of varying sizes and tested zero-shot transfer on all five test sets. The success rates of using the *Relaxed* and *Constrained* edge-matching conditions are depicted in Figure 5.3c and Figure 5.3d, respectively. We observed that LTL-Transfer using the *Relaxed* edge-matching condition successfully transferred to significantly more novel specifications across all types, thus supporting **H2**.

Relative Difficulty of Specification Types: Figure 5.3c shows that LTL-Transfer with the *Relaxed* edge-matching condition can perfectly transfer to novel tasks of *Soft*, *Strictly-Soft* and *No-Orders* types after training on very few tasks. After training on 50 tasks, LTL-Transfer can transfer to over 95% of novel tasks of the *Mixed* type. Tasks of the *Hard* type are the most difficult to transfer to. Figure 5.3d shows that the different tasks are equally difficult to transfer to using the *Constrained* edge-matching condition. Thus, **H3** is supported only by using the *Relaxed* edge-matching condition.

Stochastic Environments: To evaluate the robustness of LTL-Transfer to stochastic transitions, we increased the slip probability from 0 to 0.4 and observed a slight 3% decrease in transfer success in Figure 5.3e, which supports **H4**.

5.6.4 Robot Demonstrations

We deployed LTL-Transfer at the task-planning level of a quadruped mobile manipulator, Spot [132], with off-the-shelf motion planning and grasping capabilities in a discretized indoor environment where the robot can fetch and deliver objects while navigating through the space. LTL-Transfer first learned policies from 20 training tasks, then transferred the learned skills to 100 novel tasks from the five specification types defined in Section 5.6.2, 50 of which were executed on the robot [1]. Please see the supplementary materials for all test tasks executed on the robot. The environment contains 31 grid cells, but LTL-Transfer works in larger domains, like 19×19, as presented in Section 5.6.1. The state space includes the locations of the robot and six landmarks, i.e., two desks, a couch, a door,

[1]Videos are at `https://jasonxyliu.github.io/LTL-Transfer`

a bookshelf with a book, and a kitchen counter with a juice bottle on top. Actions are moving in the four cardinal directions. Only navigation skills are learned from the training tasks. Picking is executed when the goal location of a navigation option is the bookshelf or the counter. Placement is executed after the robot stops at the desk or the couch and has picked up an object.

5.7 Conclusion

We introduced LTL-Transfer, a zero-shot transfer algorithm that uses the compositionality of LTL task specification to maximally transfer learned policies to solve various novel tasks. Experiments in deterministic and stochastic Minecraft-inspired domains showed that LTL-Transfer can solve complex unseen tasks without violating any safety constraints. We deployed LTL-Transfer at the task-planning level of a mobile manipulator to safely solve fetch-and-deliver and navigation tasks in zero-shot. We envision incorporating long-term planning and intra-option policy updates to produce not just satisfying but optimal solutions to novel tasks.

5.8 Additional Details

5.8.1 The Implementation Details of LPOPL

LPOPL [172]'s DQN policy consists of a feedforward network with two hidden layers, each of which has 64 ReLU units. We used the same hyperparameters suggested in [172] for training, i.e., the learning rate is 0.0001; the size of the replay buffer is 25,000; 32 transitions are randomly sampled from the replay buffer for every update; the discount factor is 0.9; exploration decreases linearly from 1 to 0.02.

5.8.2 Example LTL Formulas

We provide LTL task specifications and their interpretations from the *Hard*, *Soft*, *Strictly Soft*, *No Orders*, and *Mixed* formula types.

Hard: Example formulas and their interpretations from the *Hard* type are as follows:

1. $\mathbf{F}workbench \wedge \mathbf{F}factory \wedge \mathbf{F}iron \wedge \mathbf{F}shelter \wedge \neg factory\ \mathbf{U}\ axe$: Visit *workbench*, $factory$, *iron*, *shelter*, and *axe*. Ensure that $factory$ is not visited before *axe*.

2. $\mathbf{F}toolshed \wedge \mathbf{F}bridge \wedge \mathbf{F}factory \wedge \mathbf{F}axe \wedge \neg bridge\ \mathbf{U}\ wood$: Visit *toolshed*, *bridge*, $factory$, *axe*, and *wood*. Ensure that *bridge* is not visited before *wood*.

3. $\mathbf{F}wood \wedge \mathbf{F}axe \wedge \neg wood\ \mathbf{U}\ grass \wedge \neg grass\ \mathbf{U}\ workbench \wedge \neg workbench\ \mathbf{U}\ bridge$: Visit *bridge*, *workbench*, $grass$, *wood*, and *axe*. Ensure visiting *bridge*, *workbench*, $grass$, and *wood* in that particular order. Objects that occur later in the sequence cannot be visited before any prior objects.

Soft: Example formulas and their interpretations from the *Soft* type are as follows:

1. $\mathbf{F}(bridge \wedge \mathbf{F}(factory \wedge \mathbf{F}(iron \wedge \mathbf{F}shelter)))$: Visit *bridge*, $factory$, *iron*, and

shelter in that sequence. The objects that occur later in the sequence may be visited before the prior objects, provided that they are visited at least once after the prior object has been visited.

2. $\mathbf{F}workbench \wedge \mathbf{F}(factory \wedge \mathbf{F}grass)$: Visit the *workbench*, *factory*, and *grass*. Visit *grass* at least once after visiting the *factory*.

3. $\mathbf{F}(axe \wedge \mathbf{F}factory) \wedge \mathbf{F}workbench$: Visit *axe*, *factory*, and *workbench*. Ensure that *factory* is visited at least once after *axe*.

Strictly Soft: Example formulas and their interpretations from the *Strictly Soft* type are identical to the *Soft* specifications, except they do not allow simultaneous satisfaction of multiple subtasks. The subtasks in the sequence must occur strictly after the prior subtask. This is enforced using nested operators next and finally $\mathbf{XF}a$ instead of $\mathbf{F}a$.

No Orders: These specifications only contain a list of subtasks to be completed. No temporal orders are enforced between any two subtasks.

1. $\mathbf{F}wood \wedge \mathbf{F}grass \wedge \mathbf{F}stone$: Collect *wood*, *grass*, *stone* in no particular order.

Mixed: Example formulas and their interpretations from the *Mixed* type are as follows:

1. $\mathbf{F}toolshed \wedge \mathbf{F}factory \wedge \neg toolshed \,\mathbf{U}\, shelter \wedge \mathbf{F}(grass \wedge \mathbf{F}bridge)$: Visit the *toolshed*, *factory*, *shelter*, *grass*, and *bridge*. Ensure that *toolshed* is not visited before the *shelter* and *bridge* is visited at least once after *grass*.

2. $\mathbf{F}grass \wedge \neg grass \,\mathbf{U}\, toolshed \wedge$
   $\mathbf{F}(factory \wedge \mathbf{XF}workbench)$: Visit *grass*, *toolshed*, *factory*, and *workbench*. Ensure that *grass* is not visited before *toolshed* and *workbench* is visited at least once strictly after *factory*.

3. $\mathbf{F}iron \wedge \neg iron \,\mathbf{U}\, toolshed \wedge \mathbf{F}(shelter \wedge \mathbf{XF}wood)$: Visit *iron*, *toolshed*, *shelter*, and *wood*. Ensure that *iron* is not visited before *toolshed* and *wood* is visited at least once strictly after *shelter*.

5.8.3 Additional Experimental Results

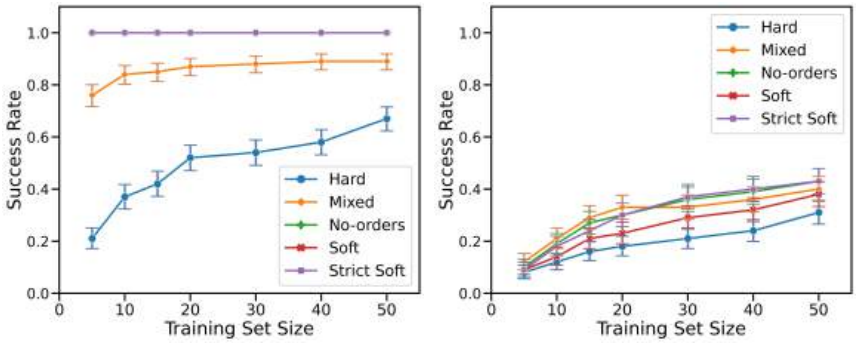
Learning Curves for Various Training Sets: We present learning curves of success rates for transferring policies learned on different specification types.

The learning curves for training on LTL tasks from the *Hard* training set with both *Relaxed* and *Constrained* edge-matching conditions are depicted in Figure 5.4.

The learning curves for training on LTL tasks from the *Soft* training set with both *Relaxed* and *Constrained* edge-matching conditions are depicted in Figure 5.5.

The learning curves for training on LTL tasks from the *Strictly Soft* training set with both *Relaxed* and *Constrained* edge-matching conditions are depicted in Figure 5.6.

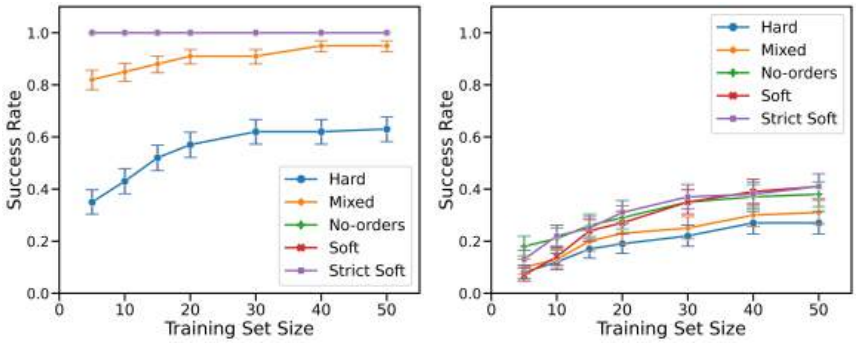
The learning curves for training on LTL tasks from the *No Orders* training set are expected to share nearly identical trends as the learning curves from the other training sets.



(a) *Relaxed* Edge Match (b) *Constrained* Edge Match

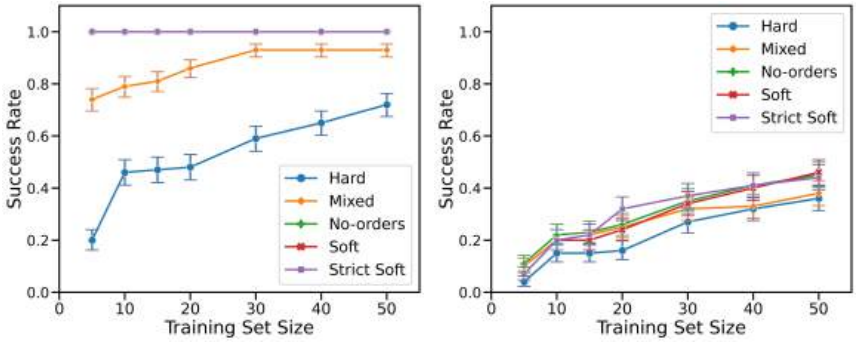
Figure 5.4: Figure 5.4a depicts the success rates of transferring to five different specifications types using the *Relaxed* edge-matching condition after LTL-Transfer being trained on *Hard* training sets of various sizes. Figure 5.4b depicts the success rates with the *Constrained* edge-matching condition. Note that the error bars depict the 95% credible interval if the successful transfer was modeled as a Bernoulli distribution.

Note that for training on each specification type, the learning curve trends are nearly identical to the learning curves of training on the *Mixed* specification types, as depicted in Figure 3 in the main paper. *Hard* specification types remain the most challenging specification types to transfer to.



(a) *Relaxed* Edge Match (b) *Constrained* Edge Match

Figure 5.5: Figure 5.5a depicts the success rates of transferring to five different specifications types using the *Relaxed* edge-matching condition after LTL-Transfer being trained on *Soft* training sets of various sizes. Figure 5.5b depicts the success rates with the *Constrained* edge-matching condition. Note that the error bars depict the 95% credible interval if the successful transfer was modeled as a Bernoulli distribution.

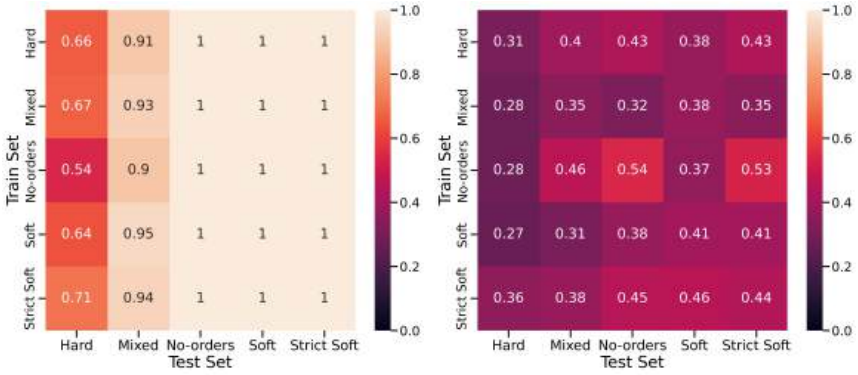


(a) *Relaxed* Edge Match (b) *Constrained* Edge Match

Figure 5.6: Figure 5.6a depicts the success rates of transferring to five different specifications types using the *Relaxed* edge-matching condition after LTL-Transfer being trained on *Strictly Soft* training sets of various sizes. Figure 5.6b depicts the success rates with the *Constrained* edge-matching condition. Note that the error bars depict the 95% credible interval if the successful transfer was modeled as a Bernoulli distribution.

Transferability of Specification Types: We evaluated whether policies learned on different specification types are more capable of transferring to all other specification types. Figure 5.7a depicts the heatmap of success rates obtained by training on 50 specifications of the type indicated by the row and transferring to 100 specifications of the type indicated by the column while using the *Relaxed* edge-matching condition. Similarly, Figure 5.7b depicts the success rates using the *Constrained* edge-matching condition. No single specification type proved to be the best training set, thus providing evidence against

the hypothesis that training with formulas conforming to certain formula templates leads to a greater success rate when transferring to all specification types. Further, training on all specification types leads to perfect transfer performance on *Soft*, *Strictly-Soft*, and *No-Orders* test set, thus providing further evidence for **H3** in the main paper.



(a) *Relaxed* Edge Match (b) *Constrained* Edge Match

Figure 5.7: Heatmaps of success rates of LTL-Transfer using the *Relaxed* and *Constrained* edge-matching conditions, respectively, on various training and test specification types.

Failure Analysis: As described in the main paper, we logged the reason for the failure of each unsuccessful transfer attempt. There are three possible causes:

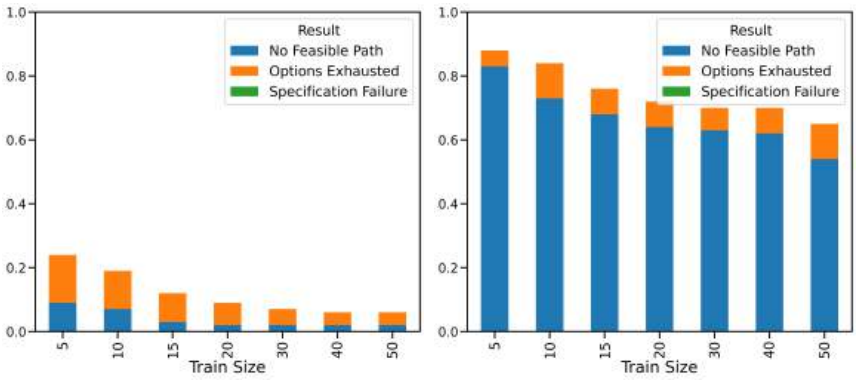
1. *Specification failure*: A constraint is violated during execution, and the reward machine progresses to an unrecoverable state.

2. *No feasible path*: After pruning the reward machine graph by removing infeasible edges, there are no paths connecting the start state to an accepting state with matching transition-centric options.

3. *Options exhausted*: During transferring, there are no further transition-centric options available to further progress the state of the reward machine.

Figure 5.8 depicts the relative frequency of the failure modes when LTL-Transfer is trained and tested on *mixed* task specifications. Note that with the *Relaxed* edge-matching condition not progressing the task after utilizing all available safe options is the primary reason for failure (Figure 5.8a) whereas, with the *Constrained* edge-matching condition,

the absence of feasible paths connecting the start and the accepting state is the primary reason for failure (Figure 5.8b). Neither has specification failures due to violating a safety constraint.



(a) *Relaxed* Edge Match (b) *Constrained* Edge Match

Figure 5.8: Reasons for failed task execution after being trained and evaluated on the *Mixed* task specification datasets. Note that all values are depicted in fractions.

5.8.4 Selected Solution Trajectories in Simulation

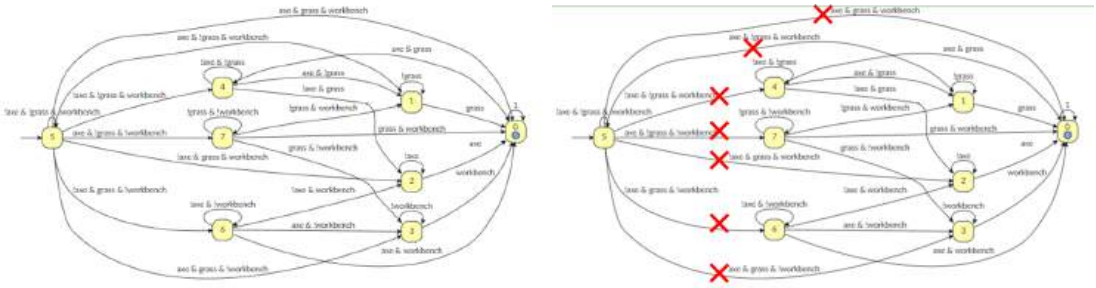
Consider the case with *Mixed* training set with 5 formulas on *Map 0*. The training formulas are:

- $\mathbf{F}grass \wedge \mathbf{F}shelter \wedge \mathbf{F}(wood \wedge \mathbf{XF}workbench)$
- $\mathbf{F}toolshed \wedge \mathbf{F}workbench \wedge \mathbf{F}shelter \wedge (\neg toolshed\ \mathbf{U}\ shelter) \wedge \mathbf{F}(grass \wedge \mathbf{F}bridge)$
- $\mathbf{F}toolshed \wedge \mathbf{F}(shelter \wedge \mathbf{F}(axe \wedge \mathbf{F}wood))$
- $\mathbf{F}iron \wedge \mathbf{F}(shelter \wedge \mathbf{XF}(bridge \wedge \mathbf{XF}factory))$
- $\mathbf{F}factory$

One of the *Mixed* test formulas is $\varphi_{test} = \mathbf{F}workbench \wedge \mathbf{F}grass \wedge \mathbf{F}axe$. The reward machine for this task specification is depicted in Figure 5.9a. Given the training set of formulas and the use of the *Constrained* edge-matching condition, the start reward machine state is disconnected from all downstream states as no transition-centric options

match the edge transitions, shown in Figure 5.9b. Therefore, LTL-Transfer does not attempt to solve the task and returns failure with the reason being *no feasible path*, i.e., a disconnected reward machine graph after removing infeasible edges.

If the *Relaxed* edge-matching condition is used, there are matching transition-centric options for each edge, shown in Figure 5.9a. The trajectory produced by LTL-Transfer when transferring the policies is depicted in Figure 5.10. The robot collects all three requisite resources before it terminates the task execution. Further, note that the robot passes through a grid containing *wood* as the specification does not explicitly prohibit it.



(a) (b)

Figure 5.9: Figure 5.9a depicts the reward machine graph for the task specification $\varphi_{test} = \mathbf{F}workbench \ \wedge \ \mathbf{F}grass \ \wedge \ \mathbf{F}axe$, as well as all feasible edges matched by the *Relaxed* edge-matching condition. Note that all the edges have at least one matching transition-centric option. Figure 5.9b depicts the edges that do not have a compatible transition-centric option for the *Constrained* edge-matching condition.

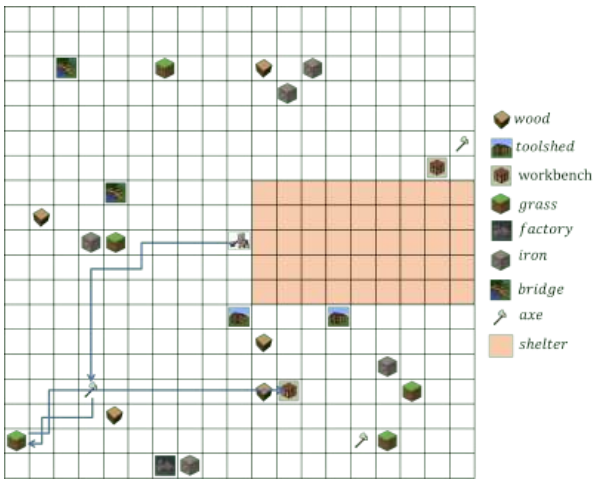


Figure 5.10: Trajectory executed by the robot using LTL-Transfer to achieve the novel task specification $\varphi_{test} = \mathbf{F}workbench \ \wedge \ \mathbf{F}grass \ \wedge \ \mathbf{F}axe$.

5.8.5 Robot Demonstration

The 50 test tasks executed on the robot are shown in Table 5.1. The Proposition a represents a brown desk, b represents a white desk, c represents a couch, d represents a door, s represents a bookshelf, and k represents a kitchen counter.

Table 5.1: Test Tasks Executed on the Robot

LTL Task Specification	Type of Task	Results
0. $\mathbf{F}a$	navigation	success
1. $\mathbf{F}a \wedge \mathbf{F}b$	navigation	success
2. $\mathbf{F}a \wedge \mathbf{F}b \wedge \mathbf{F}c$	navigation	success
3. $\mathbf{F}a \wedge \mathbf{F}b \wedge \mathbf{F}s$	navigation	success
4. $\mathbf{F}a \wedge \mathbf{F}b \wedge \mathbf{F}k$	navigation	success
5. $\mathbf{F}a \wedge \mathbf{F}b \wedge \mathbf{F}c \wedge \mathbf{F}d$	navigation	success
6. $\mathbf{F}a \wedge \mathbf{F}b \wedge \mathbf{F}c \wedge \mathbf{F}s$	navigation	success
7. $\mathbf{F}a \wedge \mathbf{F}b \wedge \mathbf{F}c \wedge \mathbf{F}k$	navigation	success
8. $\mathbf{F}a \wedge \mathbf{F}b \wedge \mathbf{F}c \wedge \mathbf{F}k \wedge \mathbf{F}s$	navigation	success
9. $\mathbf{F}(b \wedge \mathbf{F}(a \wedge \mathbf{F}(c \wedge \mathbf{F}d))))$	navigation	success
10. $\mathbf{F}(s \wedge \mathbf{F}a)$	fetch and deliver	success
11. $\mathbf{F}(s \wedge \mathbf{F}b)$	fetch and deliver	success
12. $\mathbf{F}(a \wedge \mathbf{F}b)$	navigation	success
13. $\mathbf{F}(b \wedge \mathbf{F}a)$	navigation	success
14. $\mathbf{F}(a \wedge \mathbf{F}(s \wedge \mathbf{F}c))$	fetch and deliver	success
15. $\mathbf{F}(b \wedge \mathbf{F}(s \wedge \mathbf{F}c))$	fetch and deliver	success
16. $\mathbf{F}(s \wedge \mathbf{F}(a \wedge \mathbf{F}c))$	fetch and deliver	success
17. $\mathbf{F}(a \wedge \mathbf{F}(b \wedge \mathbf{F}c))$	navigation	success
18. $\mathbf{F}(s \wedge \mathbf{F}(a \wedge \mathbf{F}(k \wedge \mathbf{F}a))))$	fetch and deliver	success
19. $\mathbf{F}(a \wedge \mathbf{F}(b \wedge \mathbf{F}(c \wedge \mathbf{F}d))))$	navigation	success
20. $\mathbf{F}(s \wedge \mathbf{XF}a)$	fetch and deliver	success
21. $\mathbf{F}(b \wedge \mathbf{XF}s)$	navigation	success
22. $\mathbf{F}(a \wedge \mathbf{XF}b)$	navigation	success
23. $\mathbf{F}(b \wedge \mathbf{XF}a)$	navigation	success
24. $\mathbf{F}(a \wedge \mathbf{XF}(b \wedge \mathbf{XF}c))$	navigation	success
25. $\mathbf{F}(a \wedge \mathbf{XF}(s \wedge \mathbf{XF}b))$	fetch and deliver	success
26. $\mathbf{F}(b \wedge \mathbf{XF}(s \wedge \mathbf{XF}a))$	fetch and deliver	success
27. $\mathbf{F}(s \wedge \mathbf{XF}(b \wedge \mathbf{XF}a))$	fetch and deliver	success
28. $\mathbf{F}(k \wedge \mathbf{XF}b)$	fetch and deliver	success
29. $\mathbf{F}(k \wedge \mathbf{XF}a)$	fetch and deliver	success
30. $\neg a\mathbf{U}s \wedge \mathbf{F}a$	fetch and deliver	success
31. $\neg b\mathbf{U}a \wedge \mathbf{F}b$	navigation	success
32. $\neg a\mathbf{U}b \wedge \mathbf{F}a$	navigation	success
33. $b\mathbf{U}a \wedge \neg c\mathbf{U}b \wedge \mathbf{F}c$	navigation	success
34. $b\mathbf{U}k \wedge \mathbf{F}b$	fetch and deliver	success
35. $\neg b\mathbf{U}c \wedge \mathbf{F}b$	navigation	success
36. $\neg a\mathbf{U}s \wedge \neg b\mathbf{U}a \wedge \mathbf{F}b$	fetch and deliver	success
37. $\neg s\mathbf{U}a \wedge \neg b\mathbf{U}s \wedge \mathbf{F}b$	fetch and deliver	success
38. $\neg b\mathbf{U}a \wedge \neg s\mathbf{U}b \wedge \mathbf{F}s$	navigation	success
39. $\neg a\mathbf{U}b \wedge \neg s\mathbf{U}a \wedge \mathbf{F}s$	navigation	success
40. $\mathbf{F}a \wedge \mathbf{F}(b \wedge \mathbf{F}c)$	navigation	success
41. $\mathbf{F}a \wedge \neg c\mathbf{U}b \wedge \mathbf{F}c$	navigation	success
42. $\mathbf{F}(a \wedge \mathbf{F}b) \wedge \neg c\mathbf{U}a \wedge \mathbf{F}c$	navigation	success
43. $\mathbf{F}a \wedge \mathbf{F}(b \wedge \mathbf{XF}c)$	navigation	success
44. $\mathbf{F}(a \wedge \mathbf{F}b) \wedge \neg c\mathbf{U}b \wedge \mathbf{F}c$	navigation	success
45. $\mathbf{F}(b \wedge \mathbf{F}a) \wedge \neg c\mathbf{U}b \wedge \mathbf{F}c$	navigation	success
46. $\mathbf{F}c \wedge (\neg s\mathbf{U}a \wedge \mathbf{F}s)$	navigation	success
47. $\mathbf{F}c \wedge (\neg a\mathbf{U}s \wedge \mathbf{F}a)$	navigation	success
48. $\mathbf{F}c \wedge (\neg s\mathbf{U}b \wedge \mathbf{F}s)$	navigation	success
49. $\mathbf{F}c \wedge (\neg b\mathbf{U}s \wedge \mathbf{F}b)$	navigation	success

CHAPTER 6

Conclusion and Future Work

Robots are becoming capable and prevalent in human environments. However, a poor task specification may induce undesired or even dangerous robot behaviors. In this thesis, we use natural language as an intuitive, expressive, and flexible way for human users to specify tasks. To solve the problem of **Robotic Language Grounding**, which addresses the challenge of connecting the words in a natural language utterance to the corresponding robot perception and actions in the physical world, we propose three questions: 1) What grounding representation to use; 2) How to ground language to the representation of choice; 3) How to produce robot actions from the grounding representation. We first provide a systematic view of the literature on robotic language grounding, propose desired properties of a grounding representation, and discuss the tradeoffs of various representations ranging from manually defined symbols to high-dimensional embeddings. By leveraging the compositionality of a structured grounding representation, linear temporal logic (LTL), we develop modular systems that ground semantically diverse spatiotemporal commands, and a transfer algorithm that uses pretrained skills to solve novel tasks while providing safety guarantees.

Robotic Language Grounding: Natural language often contains various levels of granularity. For example, a user may ask the robot to "fetch the cup and walk slowly."

This dissertation only considers using LTL to capture task-level specifications. In future work, we would like to use generalizable structures like code, keypoints, and embeddings to represent task and motion level constraints while retaining the properties that a structured representation has, e.g., compositionality, interpretability, and safety guarantees.

Human-Robot Dialogue: Natural language can be underspecified or ambiguous. For example, there may be multiple possible objects that match the word "cup" in the kitchen. However, this thesis only considers the most likely grounding. In future work, we propose explicitly modeling the probability distributions of the task and the environment representations. By quantifying these uncertainties, the robot can generate clarifying questions to query an oracle when necessary. The robot can then incorporate the response and update its belief to disambiguate the initial command.

Multimodal Grounding: Humans communicate using various modalities besides language. For example, to teach someone to drive, we use both verbal communication and demonstration while specifying visual cues. Some of the ambiguity of natural language also arises from the lack of complementary motion examples. In future work, we propose leveraging more input modalities in addition to language, including vision and demonstration. By jointly grounding all modalities, the robot can understand and solve a broader range of tasks.

Bidirectional Multimodal Human-Robot Interaction: Putting all future work mentioned above together, our ultimate goal is to develop a robotic system that can interact with people using various modalities, so we can interact and collaborate with robots just like with another person.

References

[1] Marc D. Hauser, Noam Chomsky, and W. Tecumseh Fitch. The faculty of language: What is it, who has it, and how did it evolve? *Science*, 298(5598):1569–1579, 2002.

[2] Marc D Hauser, Charles Yang, Robert C Berwick, Ian Tattersall, Michael J Ryan, Jeffrey Watumull, Noam Chomsky, and Richard C Lewontin. The mystery of language evolution. *Frontiers in psychology*, 5:88824, 2014.

[3] Maria Teresa Guasti. *Language Acquisition: The Growth of Grammar*. MIT press, 2017.

[4] Joseph Weizenbaum. ELIZA—a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45, January 1966.

[5] Artificial Intelligence Center. Shakey the robot. 1984.

[6] Stevan Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1):335–346, 1990. ISSN 0167-2789.

[7] Stevan Harnad. Symbol grounding problem. *Scholarpedia*, 2(7):2373, 2007. doi: 10.4249/scholarpedia.2373. revision #73220.

[8] 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 *IEEE International Conference on Robotics and Automation*, 2023.

[9] Michael Ahn, Anthony Brohan, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*, 2022.

[10] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. VIMA: General robot manipulation with multimodal prompts. In *International Conference on Machine Learning*, 2023.

[11] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[12] Hadas Kress-Gazit, Morteza Lahijanian, and Vasumathi Raman. Synthesis for robots: Guarantees and feedback for robot behavior. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:211–236, 2018.

[13] Andrew Ng and Michael Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in Neural Information Processing Systems*, 14, 2001.

[14] Niklas Wahlström, Thomas B Schön, and Marc Peter Deisenroth. From pixels to torques: Policy learning with deep dynamical models. *arXiv:1502.02251*, 2015.

[15] Stefanie Tellex, Nakul Gopalan, Hadas Kress-Gazit, and Cynthia Matuszek. Robots that use language. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:25–55, 2020.

[16] Ceng Zhang, Junxin Chen, Jiatong Li, Yanhong Peng, and Zebing Mao. Large language models for human-robot interaction: A review. *Biomimetic Intelligence and Robotics*, page 100131, 2023.

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

[18] 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:2401.04334*, 2024.

[19] Joy Hsu, Jiayuan Mao, Joshua B. Tenenbaum, and Jiajun Wu. What's left? concept grounding with logic-enhanced foundation models. In *Conference on Neural Information Processing Systems*, 2023.

[20] 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 (CoRL)*, 2023.

[21] Yaqi Xie, Chen Yu, Tongyao Zhu, Jinbin Bai, Ze Gong, and Harold Soh. Translating natural language to planning goals with large-language models. *arXiv:2302.05128*, 2023.

[22] 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:2304.11477*, 2023.

[23] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. ProgPrompt: Generating situated robot task plans using large language models. In *IEEE International Conference on Robotics and Automation*, 2023.

[24] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv:2305.16291*, 2023.

[25] Boyi Li, Philipp Wu, Pieter Abbeel, and Jitendra Malik. Interactive task planning with language models. In *2nd Workshop on Language and Robot Learning: Language as Grounding*, 2023.

[26] Mengjiao Yang, Yilun Du, et al. Learning interactive real-world simulators. *arXiv:2310.06114*, 2023.

[27] Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pretrained image-editing diffusion models, 2023.

[28] Yilun Du, Mengjiao Yang, Pete Florence, Fei Xia, Ayzaan Wahid, Brian Ichter, Pierre Sermanet, Tianhe Yu, Pieter Abbeel, Joshua B Tenenbaum, et al. Video language planning. *International Conference on Learning Representations*, 2024.

[29] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. PaLM-E: An embodied multimodal language model. *International Conference on Machine Learning*, 2023.

[30] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, 2023.

[31] Anthony Brohan, Noah Brown, et al. RT-1: Robotics transformer for real-world control at scale. In *Robotics: Science and Systems*, 2023.

[32] Anthony Brohan, Noah Brown, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, 2023.

[33] Abby O'Neill, Abdul Rehman, et al. Open X-Embodiment: Robotic learning datasets and RT-X models. In *IEEE International Conference on Robotics and Automation*, 2024.

[34] Jiayi Pan, Glen Chou, and Dmitry Berenson. Data-efficient learning of natural language to linear temporal logic translators for robot task specification. In *IEEE International Conference on Robotics and Automation*, 2023.

[35] Christopher Wang, Candace Ross, Yen-Ling Kuo, Boris Katz, and Andrei Barbu. Learning a natural-language to ltl executable semantic parser for grounded robotics. In *Conference on Robot Learning*, 2021.

[36] Roma Patel, Ellie Pavlick, and Stefanie Tellex. Grounding language to non-markovian tasks with no supervision of task specifications. In *Robotics: Science and Systems*, 2020.

[37] Eric Hsiung, Hiloni Mehta, Junchi Chu, Xinyu Liu, Roma Patel, Stefanie Tellex, and George Konidaris. Generalizing to new domains by mapping natural language to lifted ltl. In *IEEE International Conference on Robotics and Automation*, 2022.

[38] 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 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

[39] Francesco Fuggitti and Tathagata Chakraborti. NL2LTL – a python package for converting natural language (NL) instructions to linear temporal logic (LTL) formulas. *AAAI Conference on Artificial Intelligence*, 2023. System Demonstration.

[40] Yongchao Chen, Rujul Gandhi, Yang Zhang, and Chuchu Fan. NL2TL: Transforming natural languages to temporal logics using large language models. *arXiv:2305.07766*, 2023.

[41] Yongchao Chen, Jacob Arkin, Yang Zhang, Nicholas Roy, and Chuchu Fan. AutoTAMP: Autoregressive task and motion planning with llms as translators and checkers. In *IEEE International Conference on Robotics and Automation*, 2024.

[42] Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 152–166, 2004.

[43] Drew McDermott, Malik Ghallab, Adele E. Howe, Craig A. Knoblock, Ashwin Ram, Manuela M. Veloso, Daniel S. Weld, and David E. Wilkins. PDDL-the planning domain definition language. In *Technical Report, Tech. Rep.*, 1998.

[44] Maria Fox and Derek Long. PDDL.1: An extension to pddl for expressing temporal planning domains. *Journal of artificial intelligence research*, 20:61–124, 2003.

[45] Stefan Edelkamp and Jörg Hoffmann. PDDL2. 2: The language for the classical part of the 4th international planning competition. Technical report, Technical Report 195, University of Freiburg, 2004.

[46] Daniel L Kovacs. BNF definition of PDDL 3.1. *Unpublished manuscript from the IPC-2011 website*, 15, 2011.

[47] Malte Helmert. The fast downward planning system. *JAIR*, 26:191–246, 2006.

[48] George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Perez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61:215–289, 2018.

[49] Katherine M. Collins, Catherine Wong, Jiahai Feng, Megan Wei, and Joshua B. Tenenbaum. Structured, flexible, and robust: benchmarking and improving large language models towards more human-like behavior in out-of-distribution reasoning tasks. In *Proceedings of the 44th Annual Conference of the Cognitive Science Society*, 2022.

[50] Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning, 2023.

[51] Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. PlanBench: An extensible benchmark for evaluating large language models on planning and reasoning about change, 2023.

[52] Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. On the planning abilities of large language models : A critical investigation, 2023.

[53] Tom Silver, Soham Dan, Kavitha Srinivas, Josh Tenenbaum, Leslie Kaelbling, and Michael Katz. Generalized planning in PDDL domains with pretrained large language models. In *AAAI Conference on Artificial Intelligence*, 2024.

[54] Jake Varley, Sumeet Singh, Deepali Jain, Krzysztof Choromanski, Andy Zeng, Somnath Basu Roy Chowdhury, Avinava Dubey, and Vikas Sindhwani. Embodied ai with two arms: Zero-shot learning, safety and modularity. *arXiv:2404.03570*, 2024.

[55] Andy Zeng, Maria Attarian, brian ichter, Krzysztof Marcin Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael S Ryoo, Vikas

Sindhwani, Johnny Lee, Vincent Vanhoucke, and Pete Florence. Socratic models: Composing zero-shot multimodal reasoning with language. In *International Conference on Learning Representations*, 2023.

[56] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pages 9118–9147. PMLR, 2022.

[57] Shreyas Sundara Raman, Vanya Cohen, Eric Rosen, Ifrah Idrees, David Paulius, and Stefanie Tellex. Planning with large language models via corrective re-prompting. In *IEEE International Conference on Robotics and Automation*, 2024.

[58] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. In *Conference on Robot Learning*, 2022.

[59] Tom Brown, Benjamin Mann, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901, 2020.

[60] Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B. Brown, Prafulla Dhariwal, Scott Gray, Chris Hallacy, Benjamin Mann, Alec Radford, Aditya Ramesh, Nick Ryder, Daniel M. Ziegler, John Schulman, Dario Amodei, and Sam McCandlish. Scaling laws for autoregressive generative modeling. *arXiv:2010.14701*, 2020.

[61] Jordan Hoffmann, Sebastian Borgeaud, et al. An empirical analysis of compute-optimal large language model training. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[62] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models, 2022.

[63] Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. GAIA-1: A generative world model for autonomous driving, 2023.

[64] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-maron, Mai Giménez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. Featured Certification, Outstanding Certification.

[65] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You

Liang Tan, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems*, 2024.

[66] Wilson Yan, Danijar Hafner, Stephen James, and Pieter Abbeel. Temporally consistent transformers for video generation, 2023.

[67] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv:2304.13705*, 2023.

[68] Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, Hang Li, and Tao Kong. Vision-language foundation models as effective robot imitators. *arXiv:2311.01378*, 2023.

[69] Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (VPT): Learning to act by watching unlabeled online videos, 2022.

[70] Nakul Gopalan, Eric Rosen, George Konidaris, and Stefanie Tellex. Simultaneously learning transferable symbols and language groundings from perceptual data for instruction following. In *Robotics: Science and Systems*, 2020.

[71] Rafael Rodriguez-Sanchez, Benjamin Adin Spiegel, Jennifer Wang, Roma Patel, Stefanie Tellex, and George Konidaris. RLang: a declarative language for describing partial world knowledge to reinforcement learning agents. In *International Conference on Machine Learning*, pages 29161–29178. PMLR, 2023.

[72] Machel Reid, Nikolay Savinov, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv:2403.05530*, 2024.

[73] Nakul Gopalan, Dilip Arumugam, Lawson LS Wong, and Stefanie Tellex. Sequence-to-sequence language grounding of non-markovian task specifications. In *Robotics: Science and Systems (RSS)*, 2018.

[74] Shuang Li, Xavier Puig, Yilun Du, Clinton Wang, Ekin Akyurek, Antonio Torralba, Jacob Andreas, and Igor Mordatch. Pre-trained language models for interactive decision-making. *arXiv:2202.01771*, 2022.

[75] Richard Sutton. The bitter lesson. *Incomplete Ideas (blog)*, 13(1), 2019.

[76] Benjamin Lee Whorf. *Language, thought, and reality: Selected writings of Benjamin Lee Whorf*. MIT press, 1956.

[77] John A Lucy. *Language diversity and thought: A reformulation of the linguistic relativity hypothesis*. Cambridge University Press, 1992.

[78] David Gunning and David Aha. Darpa's explainable artificial intelligence (XAI) program. *AI magazine*, 40(2):44–58, 2019.

[79] Sule Anjomshoae, Amro Najjar, Davide Calvaresi, and Kary Främling. Explainable agents and robots: Results from a systematic literature review. In *18th Interna-*

tional Conference on Autonomous Agents and Multiagent Systems, pages 1078–1088. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[80] Andrew Silva, Mariah Schrum, Erin Hedlund-Botti, Nakul Gopalan, and Matthew Gombolay. Explainable artificial intelligence: Evaluating the objective and subjective impacts of xai on human-agent interaction. *International Journal of Human–Computer Interaction*, 39(7):1390–1404, 2023.

[81] International Electrotechnical Commission International Organization for Standardization. Functional safety of electrical/electronic/programmable electronic safety-related systems - part 1: General requirements, 2010.

[82] Vasumathi Raman, Constantine Lignos, Cameron Finucane, Kenton CT Lee, Mitchell P Marcus, and Hadas Kress-Gazit. Sorry dave, I'm afraid I can't do that: Explaining unachievable robot tasks using natural language. In *Robotics: Science and Systems*, 2013.

[83] Devleena Das, Siddhartha Banerjee, and Sonia Chernova. Explainable AI for robot failures: Generating explanations that improve user assistance in fault recovery. In *ACM/IEEE International Conference on Human-Robot Interaction*, pages 351–360, 2021.

[84] Charles Dawson, Sicun Gao, and Chuchu Fan. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control. *IEEE Transactions on Robotics*, 39(3):1749–1767, 2023. doi: 10.1109/TRO.2022.3232542.

[85] Constantine Lignos, Vasumathi Raman, Cameron Finucane, Mitchell Marcus, and Hadas Kress-Gazit. Provably correct reactive control from natural language. *Autonomous Robots*, 38:89–105, 2015.

[86] Sandeep Chinchali, Scott C Livingston, Ufuk Topcu, Joel W Burdick, and Richard M Murray. Towards formal synthesis of reactive controllers for dexterous robotic manipulation. In *IEEE International Conference on Robotics and Automation*, 2012.

[87] Ziyi Yang, Shreyas S Raman, Ankit Shah, and Stefanie Tellex. Plug in the safety chip: Enforcing constraints for LLM-driven robot agents. In *IEEE International Conference on Robotics and Automation*, 2024.

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

[89] Matthew Berg, Deniz Bayazit, Rebecca Mathew, Ariel Rotter-Aboyoun, Ellie Pavlick, and Stefanie Tellex. Grounding Language to Landmarks in Arbitrary Outdoor Environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[90] Nakul Gopalan, Dilip Arumugam, Lawson Wong, and Stefanie 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.

[91] Roma Patel, Ellie Pavlick, and Stefanie Tellex. Grounding language to non-markovian tasks with no supervision of task specifications. In *Robotics: Science and Systems*, 2020.

[92] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. *arXiv preprint arXiv:2209.11302*, 2022.

[93] Dhruv Shah, Błażej Osiński, Sergey 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.

[94] Shreyas Sundara Raman, Vanya Cohen, Eric Rosen, Ifrah Idrees, David Paulius, and Stefanie Tellex. Planning with large language models via corrective re-prompting. *arXiv preprint arXiv:2211.09935*, 2022.

[95] Michael L Littman, Ufuk Topcu, Jie Fu, Charles Isbell, Min Wen, and James MacGlashan. Environment-independent task specifications via gltl. *arXiv preprint arXiv:1704.04341*, 2017.

[96] Alberto Camacho, Rodrigo Toro Icarte, Toryn Q Klassen, Richard Anthony Valenzano, and Sheila A McIlraith. Ltl and beyond: Formal languages for reward function specification in reinforcement learning. In *IJCAI*, volume 19, pages 6065–6073, 2019.

[97] Yoonseon Oh, Roma Patel, Thao Nguyen, Baichuan Huang, Ellie Pavlick, and Stefanie Tellex. Planning with State Abstractions for Non-Markovian Task Specifications. In *Proceedings of Robotics: Science and Systems*, Freiburg, Germany, June 2019.

[98] Rodrigo Toro Icarte, Toryn Q Klassen, Richard Valenzano, and Sheila A McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research*, 73:173–208, 2022.

[99] Jason Xinyu Liu, Ankit Shah, Eric Rosen, Mingxi Jia, George Konidaris, and Stefanie Tellex. Skill transfer for temporally-extended task specifications. *arXiv preprint arXiv:2206.05096*, 2022.

[100] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[101] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[102] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language Models are Unsupervised Multitask Learners. *OpenAI blog*, 1(8):9, 2019.

[103] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda

Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[104] Josh Achiam, Steven Adler, et al. GPT-4 technical report, 2023.

[105] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter 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.

[106] Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila 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.

[107] Giuseppe De Giacomo, Luca Iocchi, Marco Favorito, and Fabio 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.

[108] Ankit Shah, Shen Li, and Julie Shah. Planning with uncertain specifications (PUnS). *IEEE Robotics and Automation Letters (RA-L)*, 5(2):3414–3421, 2020.

[109] Ankit Shah, Pritish Kamath, Julie A Shah, and Shen Li. Bayesian inference of temporal task specifications from demonstrations. *Advances in Neural Information Processing Systems*, 31, 2018.

[110] Marcell Vazquez-Chanlatte, Susmit Jha, Ashish Tiwari, Mark K Ho, and Sanjit Seshia. Learning task specifications from demonstrations. In *Advances in Neural Information Processing Systems 31*, pages 5368–5378. 2018.

[111] Joseph Kim, Christian Muise, Ankit Shah, Shubham Agarwal, and Julie Shah. Bayesian inference of linear temporal logic specifications for contrastive explanations. In *IJCAI*, 2019.

[112] Ankit Shah, Samir Wadhwania, and Julie Shah. Interactive robot training for non-markov tasks. *arXiv preprint arXiv:2003.02232*, 2020.

[113] 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.

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

[115] Rob Gerth, Doron Peled, Moshe Y Vardi, and Pierre 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.

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

[117] Matt MacMahon, Brian Stankiewicz, and Benjamin 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.

[118] David L. Chen and Raymond 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.

[119] Joohyun Kim and Raymond 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.

[120] Yoav Artzi and Luke Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62, 2013.

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

[122] Matthias Cosler, Christopher Hahn, Daniel Mendoza, Frederik Schmitt, and Caroline Trippel. nl2spec: Interactively translating unstructured natural language to temporal logics with large language models. *arXiv preprint arXiv:2303.04864*, 2023.

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

[124] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. 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, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1154.

[125] John Lyons. *Semantics: Volume 2*, volume 2. Cambridge university press, 1977.

[126] Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, et al. Text

and code embeddings by contrastive pre-training. *arXiv preprint arXiv:2201.10005*, 2022.

[127] Matthew Berg, Deniz Bayazit, Rebecca Mathew, Ariel Rotter-Aboyoun, Ellie Pavlick, and Stefanie Tellex. Grounding Language to Landmarks in Arbitrary Outdoor Environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[128] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander 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, October 2020. Association for Computational Linguistics.

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

[130] Alexandre Duret-Lutz, Etienne Renault, Maximilien Colange, Florian Renkin, Alexandre Gbaguidi Aisse, Philipp Schlehuber-Caissier, Thomas Medioni, Antoine Martin, Jérôme Dubois, Clément Gillard, and Henrich Lauko. From Spot 2.0 to Spot 2.10: What's new? In *Proceedings of the 34th International Conference on Computer Aided Verification (CAV'22)*, volume 13372 of *Lecture Notes in Computer Science*, pages 174–187. Springer, August 2022. doi: 10.1007/978-3-031-13188-2_9.

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

[132] Boston Dynamics. Spot® - The Agile Mobile Robot. `https://www.bostondynamics.com/products/spot`.

[133] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018.

[134] Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-Across-Room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Conference on Empirical Methods for Natural Language Processing*, 2020.

[135] Kaiyu Zheng, Deniz Bayazit, Rebecca Mathew, Ellie Pavlick, and Stefanie Tellex. Spatial language understanding for object search in partially observed city-scale environments. In *IEEE International Conference on Robot & Human Interactive Communication*, pages 315–322, 2021.

[136] Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12538–12547, 2019.

[137] Dhruv Shah, Błażej Osiński, Sergey 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, 2023.

[138] Jason Xinyu Liu, Ziyi Yang, Ifrah Idrees, Sam Liang, Benjamin Schornstein, Stefanie Tellex, and Ankit Shah. Lang2LTL: Grounding complex natural language commands for temporal tasks in unseen environments. In *Conference on Robot Learning*, pages 1084–1110. PMLR, 2023.

[139] Michael L Littman, Ufuk Topcu, Jie Fu, Charles Isbell, Min Wen, and James MacGlashan. Environment-independent task specifications via GLTL. *arXiv preprint arXiv:1704.04341*, 2017.

[140] Alberto Camacho, Rodrigo Toro Icarte, Toryn Q Klassen, Richard Anthony Valenzano, and Sheila A McIlraith. LTL and beyond: Formal languages for reward function specification in reinforcement learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, volume 19, pages 6065–6073, 2019.

[141] Yoonseon Oh, Roma Patel, Thao Nguyen, Baichuan Huang, Ellie Pavlick, and Stefanie Tellex. Planning with state abstractions for non-markovian task specifications. In *Robotics: Science and Systems (RSS)*, volume 2019, 2019.

[142] Rodrigo Toro Icarte, Toryn Q Klassen, Richard Valenzano, and Sheila A McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research (JAIL)*, 73:173–208, 2022.

[143] Jason Xinyu Liu, Ankit Shah, Eric Rosen, Mingxi Jia, George Konidaris, and Stefanie Tellex. LTL-Transfer: Skill transfer for temporal task specification. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[144] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter 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.

[145] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[146] Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. Vision-language models for vision tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–20, 2024.

[147] Timo Lüddecke and Alexander Ecker. Image segmentation using text and image

prompts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7086–7096, 2022.

[148] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, et al. Simple open-vocabulary object detection. In *European Conference on Computer Vision*, pages 728–755. Springer, 2022.

[149] Jun Chen, Han Guo, Kai Yi, Boyang Li, and Mohamed Elhoseiny. VisualGPT: Data-efficient adaptation of pretrained language models for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18030–18040, 2022.

[150] Zheyuan Liu, Cristian Rodriguez-Opazo, Damien Teney, and Stephen Gould. Image retrieval on real-life images with pre-trained vision-and-language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2125–2134, 2021.

[151] Yifan Du, Junyi Li, Tianyi Tang, Wayne Xin Zhao, and Ji-Rong Wen. Zero-shot visual question answering with language model feedback. In *Findings of the Association for Computational Linguistics*, pages 9268–9281. Association for Computational Linguistics, 2023.

[152] Zhengyuan Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. The dawn of LMMs: Preliminary explorations with GPT-4V(ision). *arXiv preprint arXiv:2309.17421*, 2023.

[153] Matthew Berg, Deniz Bayazit, Rebecca Mathew, Ariel Rotter-Aboyoun, Ellie Pavlick, and Stefanie Tellex. Grounding language to landmarks in arbitrary outdoor environments. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 208–215. IEEE, 2020.

[154] Ankit Shah, Pritish Kamath, Shen Li, Patrick Craven, Kevin Landers, Kevin Oden, and Julie Shah. Supervised bayesian specification inference from demonstrations. *The International Journal of Robotics Research*, 42(14):1245–1264, 2023.

[155] Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila 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.

[156] Giuseppe De Giacomo, Luca Iocchi, Marco Favorito, and Fabio 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.

[157] Barbara Landau and Ray Jackendoff. "what" and "where" in spatial language and spatial cognition. *Behavioral and Brain Sciences*, 16:217–265, 06 1993. doi: 10.1017/S0140525X00029733.

[158] Victor Weixin Liang, Yuhui Zhang, Yongchan Kwon, Serena Yeung, and James Y

Zou. Mind the gap: Understanding the modality gap in multi-modal contrastive representation learning. *Advances in Neural Information Processing Systems*, 35: 17612–17625, 2022.

[159] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, 2020.

[160] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[161] OpenStreetMap Contributors. Planet OSM. `https://www.openstreetmap.org`, 2017.

[162] Dragomir Anguelov, Carole Dulong, Daniel Filip, Christian Frueh, Stéphane Lafon, Richard Lyon, Abhijit Ogale, Luc Vincent, and Josh Weaver. Google Street View: Capturing the world at street level. *Computer*, 43(6):32–38, 2010.

[163] Vanya Cohen, Jason Xinyu Liu, Raymond Mooney, Stefanie Tellex, and David Watkins. A survey of robotic language grounding: Tradeoffs between symbols and embeddings. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2024.

[164] Junha Roh, Karthik Desingh, Ali Farhadi, and Dieter Fox. LanguageRefer: Spatial-language model for 3d visual grounding. In *Conference on Robot Learning*, pages 1046–1056. PMLR, 2022.

[165] Sourav Garg, Krishan Rana, Mehdi Hosseinzadeh, Lachlan Mares, Niko Suenderhauf, Feras Dayoub, and Ian Reid. RoboHop: Segment-based topological map representation for open-world visual navigation. In *IEEE International Conference on Robotics and Automation*. IEEE, 2024.

[166] Hadas Kress-Gazit, Georgios E Fainekos, and George J Pappas. From structured english to robot motion. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2717–2722. IEEE, 2007.

[167] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to Sequence Learning with Neural Networks. *Advances in Neural Information Processing Systems*, 27, 2014.

[168] Benedict Quartey, Eric Rosen, Stefanie Tellex, and George Konidaris. Verifiably following complex robot instructions with foundation models. *arXiv preprint arXiv:2402.11498*, 2024.

[169] Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Visual language

maps for robot navigation. In *IEEE International Conference on Robotics and Automation*, pages 10608–10615. IEEE, 2023.

[170] M.E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

[171] Michael L Littman, Ufuk Topcu, Jie Fu, Charles Isbell, Min Wen, and James MacGlashan. Environment-independent task specifications via GLTL. *arXiv:1704.04341*, 2017.

[172] Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. Teaching multiple tasks to an RL agent using LTL. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 452–461, 2018.

[173] Ankit Shah, Pritish Kamath, Julie A Shah, and Shen Li. Bayesian inference of temporal task specifications from demonstrations. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

[174] Glen Chou, Dmitry Berenson, and Necmiye Ozay. Learning constraints from demonstrations with grid and parametric representations. *The International Journal of Robotics Research (IJRR)*, 40(10-11):1255–1283, 2021.

[175] Glen Chou, Necmiye Ozay, and Dmitry Berenson. Learning temporal logic formulas from suboptimal demonstrations: theory and experiments. *Autonomous Robots*, 46 (1):149–174, 2022.

[176] Rob Gerth, Doron Peled, Moshe Y Vardi, and Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *International Conference on Protocol Specification, Testing and Verification*, pages 3–18. Springer, 1995.

[177] Orna Kupferman and Moshe Y Vardi. Model checking of safety properties. *Formal methods in system design*, 19(3):291–314, 2001.

[178] Zohar Manna and Amir Pneuli. A hierarchy of temporal properties. In *ACM symposium on Principles of distributed computing*, 1990.

[179] Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211, 1999.

[180] Kishor Jothimurugan, Suguman Bansal, Osbert Bastani, and Rajeev Alur. Compositional reinforcement learning from logical specifications. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[181] Borja G León, Murray Shanahan, and Francesco Belardinelli. Systematic generalisation through task temporal logic and deep reinforcement learning. *Adaptive and Learning Agents Workshop at International Conference on Autonomous Agents and Multiagent Systems*, 2022.

[182] Borja G Leon, Murray Shanahan, and Francesco Belardinelli. In a nutshell, the

human asked for this: Latent goals for following temporal specifications. In *International Conference on Learning Representations (ICLR)*, 2022.

[183] Brandon Araki, Xiao Li, Kiran Vodrahalli, Jonathan Decastro, Micah Fry, and Daniela Rus. The logical options framework. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 307–317. PMLR, 18–24 Jul 2021.

[184] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning*, pages 166–175. PMLR, 2017.

[185] Yen-Ling Kuo, Boris Katz, and Andrei Barbu. Encoding formulas as deep networks: Reinforcement learning for zero-shot execution of ltl formulas. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5604–5610. IEEE, 2020.

[186] Pashootan Vaezipoor, Andrew C Li, Rodrigo A Toro Icarte, and Sheila A Mcilraith. LTL2action: Generalizing LTL instructions for multi-task RL. In *International Conference on Machine Learning*, pages 10497–10508. PMLR, 2021.

[187] Zhe Xu and Ufuk Topcu. Transfer of temporal logic formulas in reinforcement learning. In *IJCAI: proceedings of the conference*, volume 28, page 4010. NIH Public Access, 2019.

[188] Geraud Nangue Tasse, Devon Jarvis, Steven James, and Benjamin Rosman. Skill machines: Temporal logic composition in reinforcement learning. *arXiv preprint arXiv:2205.12532*, 2022.

[189] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[190] Thomas M Moerland, Joost Broekens, Aske Plaat, Catholijn M Jonker, et al. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1):1–118, 2023.

[191] George Dimitri Konidaris and Andrew G Barto. Building portable options: Skill transfer in reinforcement learning. In *IJCAI*, volume 7, pages 895–900, 2007.

[192] Steven James, Benjamin Rosman, and George Konidaris. Learning portable representations for high-level planning. In *International Conference on Machine Learning*, pages 4682–4691. PMLR, 2020.

[193] Akhil Bagaria and George Konidaris. Option discovery using deep skill chaining. In *International Conference on Learning Representations*, 2019.

[194] Akhil Bagaria, Jason Senthil, Matthew Slivinski, and George Konidaris. Robustly learning composable options in deep reinforcement learning. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, 2021.

[195] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[196] Leslie G Valiant. The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201, 1979.

[197] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, January 2017.