Identification of Atomic Propositions in English Instructions for Flexible Translation to Robot Planning Representations

by

Rujul Gandhi

S.B. Linguistics & Philosophy and Electrical Engineering & Computer Science, Massachusetts Institute of Technology (2022)

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Scienc

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2023

Authored by:	Rujul Gandhi Department of Electrical Engineering and Computer Science August 11, 2023
Certified by:	Chuchu Fan Assistant Professor Thesis Supervisor
Certified by:	Yang Zhang Research Scientist, MIT-IBM Watson AI Lab Thesis Supervisor
Accepted by:	Katrina LaCurts Chair, Master of Engineering Thesis Committee

Identification of Atomic Propositions in English Instructions for Flexible Translation to Robot Planning Representations

by

Rujul Gandhi

Submitted to the Department of Electrical Engineering and Computer Science
on August 11, 2023, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Scienc

Abstract

Creating human-interactive problem-solving robots involves interfacing natural-language instructions into formal representations. This formal representation should contain all the verifiable constituent units (ideally atomic propositions) which are present in the natural language instruction. However, the format and vocabulary of atomic propositions may vary substantially across formal representations and their application domains. Hence, extracting the correct atomic propositions from natural language has been a bottleneck in converting language to formal representations. In this thesis, we propose and implement a two-step method for identifying atomic propositions in a representation-agnostic way. Given an instruction in natural English, we first identify the spans of that instruction that may potentially be atomic propositions, and then carry out a finer-grained translation into the chosen formalization language. In evaluating this approach, we demonstrate the ability of the span identification method to generalize to two common domains of robot planning tasks, navigation and manipulation, as well as three additional domains of household robot tasks. Finally, we discuss, implement, and evaluate methods to incorporate span identification into the process of parsing English into three formal representations: Temporal Logic, PDDL, and a custom style of atomic propositions. Using pretrained language models and naturalistic parallel data, we build a system that enables flexible formalization of natural language across chosen intermediate representations.

Thesis Supervisor: Chuchu Fan
Title: Assistant Professor

Thesis Supervisor: Yang Zhang
Title: Research Scientist, MIT-IBM Watson AI Lab

Acknowledgments

The past year has been both tumultuous and rewarding. I've been fortunate to have so many sources of support that made this journey lighter.

I'm grateful to my thesis supervisor, Chuchu Fan, for valuable advice about how to approach research, as well as the rest of the Realm lab for always being welcoming and available. Thank you to my industry supervisor Yang Zhang for helping me problem-solve and being a voice of reason as I tried to untangle my freshly tossed thoughts every now and then. I'm grateful to Jake Arkin for insightful advice and being willing to step in when I most needed guidance.

There is a world outside of my academic life, without which my academic life would not be the same. I am grateful to the MIT Lightweight Women for being an inspiring community and unwavering presence to come back to every day. In particular, thank you to Nicole for thesis-writing solidarity, and Coach Amelia Patton for being overall awesome. I'm grateful to every single member of my living community, pika, for listening patiently and being my greatest source of joy. Never have I appreciated my friends more than I do now. Thank you to Shardul for endless discussions, laughs, listening, and believing in me every step of the way. Thank you to friends who I could always rely on at MIT - Shinjini, Sualeh, Shriya, Stuti, Shreya - and those who lent timely support during the past year, particularly Aalok, Vikram, and Siddhartha.

This and everything else I do is due to my family: Dada, Aai, and Sanmay, who have always been there on the days when I needed it the most. Thank you for reminding me what truly matters.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The ability of humans to interact with robotic machines using language is at a stage of rapid development. Phones come equipped with virtual assistants that 'listen' to natural language queries and instructions, and smart-home devices turn appliances on and off with just a verbal command. But a lot of nuance is missing, and it becomes important as we start using language to describe more complex tasks and problems. Although we use language seamlessly, our everyday utterances are full of ambiguity, hidden implications, and complex structures that may be opaque to robotic systems. In a world where any user can communicate their goals to a robotic machine with ease, complete and scalable natural language understanding is a crucial part of the human-robot interface.

Imagine a robot in a lab, carrying out an impressive demonstration of planning based on a natural language instruction. A human says, *"There is a red block on the table. Grasp the block until the light turns on and then pick it up."* This instruction is executed. Impressive! Now, imagine we want to scale this machine's abilities into a variety of domains, some wildly different. What ought to happen when a human says, *"There is a cat stuck in the stairwell of Building 36! Search the stairwell until you see her and then send us a message."*?

Although these sentences describe very different situations and potentially require very different action repertoires from the robot, they are similar at the natural language understanding level. If, instead of trying to parse language directly into a set of actions, a robot was trying to first build up a formal definition of the desired task, we would find that these two very different commands can be structured exactly the same way in that intermediate representation. The specific actions from the robot's repertoire only need to be plugged in to the intermediate representation.

This thesis takes a step towards better constructing these intermediate representations from natural language. Specifically, we ask: how can a machine correctly identify sub-tasks or sub-constraints from a complex utterance in natural language, so that their interdependencies may be formally described? We hope that the resulting system makes it easier to leverage the structure of human language into making machine-language interfaces more generalizable and interpretable.

1.1 Motivation

Communicating a task to a machine typically requires precisely defining the problem, constraints, and goals in a formal representation. Formal representations of tasks and goals facilitate algorithmic planning, increase the efficiency of verification methods, and help us build robust and reliable systems as a result. In natural language, a person can describe their desired tasks or goal states to a robot by giving it an instruction. This instruction could be complicated or ambiguous. It could be composed of subtasks or sub-states that interact. A formal representation of the same task would consist of smaller verifiable units (ideally atomic propositions) corresponding to all of the sub-tasks, linked together using the appropriate operators or syntax as defined by the formalization language. Figure 1-1 shows an example of breaking down sub-tasks in order to represent the complex logic of an instruction.

The task of identifying these atomic propositions in natural language instructions has cross-domain applications. In the field of converting natural language to temporal logic, difficulty in correctly identifying and grounding atomic propositions has been cited as a factor that prevents or limits generalization to new domains [14]. Decomposing a user's instruction into smaller, focused units is also a strategy for planning from high-level instructions [1] and providing targeted feedback to the user in case of

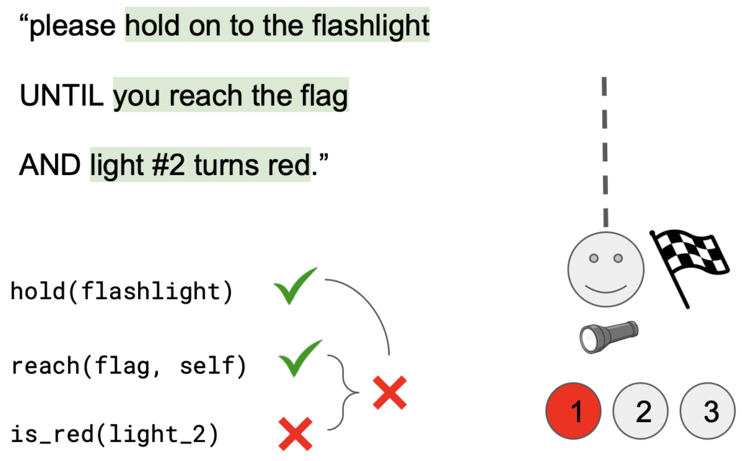


Figure 1-1: An example of breaking down a natural language instruction into atomic propositions. The APs, highlighted in the instruction on top left, are treated as individual logical units (bottom left) in order to evaluate whether the robot needs to be holding the flashlight at the given point of time.

system failure [25].

1.2 Problem Statement

We are given an instruction in natural language (NL). First, we *assume that the semantics of the instruction can be represented using a logical formula.*

Such a formula consists of atomic propositions (APs) which evaluate to Boolean values (T/F). These APs are put together by operators from a finite set, which might be defined through a framework such as propositional logic or temporal logic.

If the NL instruction is a sequence of tokens $w_1, w_2, w_3 ... w_n$ then each token will belong to one of two categories – either part of an operator, or part of an atomic proposition.

At the end, we want to make an ordered list of non-overlapping subsets, such that each subset can be parsed into an atomic proposition.

The remainder of the token sequence will be assumed to contain the operators. We aim to identify the subsets of tokens that correspond to atomic propositions in a way that is agnostic to the specific operators present in the formalization framework.

1.3 Project

The project of this thesis is building a modular system to identify atomic propositions from a natural language instruction. First, we fine-tune a pre-trained language model to identify the spans of an instruction corresponding to potential atomic propositions. Then, we use the identified spans as an input into whatever formalization framework is desired. To demonstrate the use of the identified spans as input to a formalization framework, we implement a conversion into Planning Domain Definition Language (PDDL) as well as a conversion into atomic propositions represented in a custom style (e.g. `predicate(object)`).

1.3.1 Goals

Dataset creation Our methodological goal is to create a dataset for extracting potential atomic propositions from English commands and release it for further use. In creating the dataset, we use a Large Language Model (LLM) as a data augmentation tool, saving hours of human annotation time. When releasing the dataset, we provide a detailed description and evaluation of the data augmentation technique. We also demonstrate that the dataset thus created is more diverse, natural, and complex than existent algorithmically augmented datasets in this field.

Pretrained networks for language-to-TL conversion Our engineering goal is to demonstrate the incorporation of pretrained networks into the task of identifying atomic propositions. Building upon the idea that starting with some sort of abstract knowledge representation can help us generalize better in a downstream task, we demonstrate this for the identification of constraints from natural language sentences spanning five different domains.

Multifaceted evaluation Our scientific goal is evaluation of every step of our pipeline. Within each layer, we enumerate the design decisions and carry out tests on a variety of metrics to identify the effects of that decision on the final usefulness of the model. In addition to evaluating on test set accuracy, we identify past pain points – generalization ability, validity of LLM generations, and interpretability – and design experiments to evaluate our approach on these metrics. We hope that this multi-pronged evaluation approach, going beyond benchmark accuracy, can be adapted into other domains and highlight areas for dramatic improvement in the current state of the art.

This project contributes to a broader research theme of parsing constraints expressed in natural language into a formal intermediate representation.

1.4 Contributions

To summarize, there are three key contributions in this thesis:

1. Demonstrate the incorporation of modern NLP techniques into the longstanding problem of extracting atomic propositions from linguistic input.

2. Create a novel dataset for AP identification which is diverse, complex, and natural.

3. Evaluate the system on accuracy and generalizability. Further, evaluate the merit of the two-step system that breaks down NL instructions prior to formalization.

1.5 Thesis Overview

The reminader of the thesis is structured as follows. Chapter 2 provides the technical background of relevant topics such as atomic propositions, temporal logic, and PDDL. Chapter 3 provides an overview of related work in the language-to-formalization field, with a focus on temporal logic as a well-studied formalization.

Chapters 4 and 5 focus on the technical methods of building the system. In Chapter 4, we focus on the span identification aspect of the model. In Chapter 5, we focus on the LLM-assisted methods of actually grounding the actionable spans to actions or propositions in the robot's real-world environment. In both of these chapters, we describe the experiments used to evaluate the technical approach and share the results.

Finally, Chapter 6 provides a summary of findings and additional notes, along with a discussion of limitations and future directions to this work.

Chapter 2

Technical Background

In this chapter, we review the logical representations that are relevant to this work.

2.1 Atomic Propositions

In logic, a proposition is a statement that can be evaluated as True or False, for instance, $p \rightarrow q$ (p implies q). An atomic proposition is one where further decomposition is not possible. In the above example, p and q would be considered atomic propositions. In this paper, we use the term 'atomic proposition' to mean a minimal unit that can be verified (it is either a checkable state, or an action that can be completed or not completed) and further breakdown is not *required.*

2.2 Temporal Logic

In temporal logic, atomic propositions representing particular constraints or tasks are linked together by temporal operators as well as standard propositional logic operators [22]. It is used in situations where a robotic machine needs to satisfy certain constraints that are interdependent and *ordered in time.* For instance, Linear Temporal Logic (LTL) is a type of temporal logic which, in addition to propositional logic operators, contains the additional operators **F** (at some point in the **F**uture), **U** (**U**ntil), and **G** (**G**lobally). An example of an LTL statement is below. Each Φ_i is an atomic proposition.

$F\ (\ \varphi_1\ U\ (\ \varphi_2\ \&\ \varphi_3\ )\ )$

At some point in the Future, Φ_1 must be true Until (Φ_2 And Φ_3 are true).

e.g. "(hold on to the flashlight)$_1$ until (you reach the flag)$_2$ and (light #2 turns red)$_3$"

2.3 PDDL

The Planning Domain Definition Language [20], or PDDL, is a Lisp-based framework for describing a problem to a robotic planner. Specifically, in PDDL the user describes the problem domain by describing available actions, predicates, and object types. The user then can describe any problem in this domain by describing the specific objects, initial conditions, and goal conditions that have to be met. Some robotics simulators automatically detect objects and initial conditions from a scene [17]. What remains to complete a problem definition is to provide the goal conditions. The goal conditions, like the initial conditions, have to be defined in terms of the objects, predicates, and actions which are present in the domain definition. So, if something similar to atomic propositions are extracted from a natural language instruction, a PDDL formalization could be constructed by converting each AP into a clause within the goal statement.

Chapter 3

Literature Review

3.1 The Language-Robot Interface

3.1.1 Early NLP for Robotics - Defining the Problem

Early exploration of the language-computer interface, much like the bulk of current research, focused on simulating human-to-human conversation. However, there were some attempts to create natural language understanding in order to interact with machines and have them carry out tasks. A demonstrative example developed in the 60s is the SHRDLU system [32]. SHRDLU operated in a limited toy setting called ‘blocksworld’, and its demonstrated abilities included understanding what objects were being referred to in a natural language instruction. It gave feedback to the user if a particular object was not understood, not existent in the environment, or a particular action was not in its repertoire.

Through six decades of work since then, the underlying desiderata of SHRDLU are not very different from what we want from our systems today. When we say that we want our systems to ‘understand’ natural language instructions, we often mean that we would like our system to interface between the instruction we have provided and a concrete instruction that it is able to carry out in its own environment – while communicating back to us if such a mapping is not possible. This highlights **interpretability** as an important and longstanding concern for the natural language understanding space.

Surprisingly, even SHRDLU's shortcomings were not too different from the key challenges with NLU for robotics today. SHRDLU was constrained to the rudimentary blocksworld environment, and only responded with reasonable success to a few well-tested structures of instructions. Today, robotics NLU systems are often tested on a single domain and fail to generalize to others [14]. This highlights another concern – **generalizability** of a natural language understanding solution.

3.1.2 Statistical Methods - Towards a Data-Centric Approach

In the 1980s, the rise of statistical methods in language processing reduced the reliance on painstakingly crafted dictionaries and grammatical rules. These methods used data to learn the particular values of a set of parameters in a pre-defined model. Use of data began to show promise as a reliable way to improve model performance [19]. A drawback of statistical methods was that as the models grew more complicated, complex feature engineering was required in order to define all the correct features to capture the different aspects of language. This is where neural networks came in.

Neural network models are nonparametric – meaning that there isn't a fixed set of parameters or features. Instead, over the course of the training phase, a neural network defines and tunes its own set of features. While this makes neural networks much more generalizable, it also makes them opaque to a user. This underscores the importance of an **explainable intermediate representation** in case of failure in these systems.

3.1.3 Pretrained Language Models - The State of NLP Today

A class of neural network models are commonly referred to as Large Language Models (LLMs). An LLM is pre-trained on a large amount - terabytes - of unlabeled data. The LLM learns patterns of language use that are then encoded into billions of parameters, enabling much finer-grained responses over a larger set of domains than ever possible before.

Pretrained language models may be trained on objectives such as next-word prediction or masked language modeling. Masked language modeling, introduced with BERT [7], makes use of unlabeled data by randomly 'masking' spans of tokens[1]. The model's objective is to choose the highest probability sequence of tokens to fill in the masked region. Loss is calculated based on the similarity of the model's output to the original sentence. The model we use in our system, T5, uses a similar masked-language-modeling method for pretraining. Pretraining for T5 is done using the Colossal Clean Crawled Corpus [24], a large corpus of cleaned English text scraped from the internet using Common Crawl (https://commoncrawl.org/).

T5 introduced the idea of treating any NLP problem as a text-to-text problem. After pretraining on a large unlabeled corpus, T5 could be fine-tuned on multiple specific NLP tasks, converted to a format in which the input was a string and the output was a string as well. The method of pre-training a model and then fine-tuning on a specific task with fewer labeled examples is called transfer learning, and originating in the field of Computer Vision, it is very common in NLP today. We will refer to transfer learning in the context of our model in later chapters.

3.2 Parsing Language into Logical Intermediates

Work from the early 2000s addressing the language-robot-interface recognizes the need for a domain-independent logical intermediate between a natural language instruction and the robot's semantic representation of the task [16]. Both prior to and following this, much natural-language-understanding work focused on predicate logic [31] or temporal logic [15] as potential intermediates due to a structure that allowed for easier rule-based translation from structured English sentences.

TL is not the only option for a logical intermediate, however. Algorithmic planners were developed to solve problems defined in PDDL, and so it provides a useful intermediate representation.

[1]A finer-grained representation than 'words', a word may consist of multiple tokens.

3.2.1 Language to Temporal Logic

[4] provides a recent (2019) survey of approaches to convert natural language, usually English, into Linear Temporal Logic (LTL) specifications. This survey identifies some of the challenges that arise in converting language, with potential ambiguities, into Temporal Logic. It divides approaches up to 2019 into two broad categories - rule-based approaches and statistical approaches - and finds that overall, statistical approaches are capable of outperforming rule-based approaches, but only when there is an abundance of data used to train the model.

3.2.2 Language to PDDL

The Planning Domain Definition Language (PDDL) is a commonly used problem definition language for robotics. Although pretrained large language models do not necessarily produce reasonable plans from natural language instructions, they can be used as translators between natural language and PDDL goal statements, and combined with algorithmic planners which rely on goals defined in PDDL [11, 33]. This approach is a current area of research.

Chapter 4

Methods: Span Identification

The first step in the two-step process to identify atomic propositions is identifying spans of the natural language instruction that might potentially contain the APs. In this section, we detail the technical implementation of this step. Then, we carry out evaluation of the span identification model on metrics of accuracy and cross-domain generalization.

4.1 Technical Implementation

The technical implementation of the span identification model involved creating a dataset and fine-tuning a T5 model on the task. The following sections describe the dataset, including the creation process and an analysis of the data distribution. Then, we talk about fine-tuning and postprocessing.

4.1.1 Dataset

A goal of the language-to-formalization effort is making formal descriptions available to a layperson. But existent datasets often lack examples of the way that a layperson might speak to a robot. Prior work in text-to-TL has relied on synthetically generating English sentences from randomly generated Temporal Logic using hard-coded rules [12, 30]. Although some randomness can be added, this approach limits the flexibility and diversity of data, thus highly limiting the generalization potential of the model trained on it [10]. Instead, we propose starting with manually annotated human-generated sentences and then augmenting this data using a large language model.

We collected about 600 labeled examples and doubled that through augmentation to build a dataset of 1.3K. We incorporate common robot toy environments in the navigation and manipulation domain as well as relatable and applicable household tasks. For our natural data, we use Wikihow[1] as the primary source. Some data is compiled from other works on robotics [27] as well as from a crowd-sourcing effort with collaborators familiar with logical representations. This initial dataset is referred to as the 'seed data' in the following section.

In terms of format, our dataset consisted of English instructions as the input and the AP-lifted instruction as well as a list of APs as the label. This allowed for a supervised training approach, in which the 'ground truth' labels are provided to the model at training time. Testing is then done on a subset of the dataset that has been withheld completely during test time. Following from work that shows that most tasks can be represented in a text-to-text format with good performance [24], we converted our dataset into a text-to-text format as shown in Figure 4-1.

4.1.2 Data Augmentation

We use GPT-3 [23], a pre-trained large language model, to augment our data. This is a departure from the precedent of rule-based data augmentation [12, 30]. It is an important one. By virtue of being pre-trrained on large amounts of human-generated data, GPT-3 encodes finer patterns of naturalistic utterances than what can possibly be hard-coded by a small team of researchers.

The data augmentation process consists of two techniques. First, we carried out **vocabulary expansion**. This meant retaining the sentence structures from our seed data but enhancing the vocabulary that they contain. In order to do this, we take the AP-lifted version of each data point. Then, we use in-context learning with GPT-3 or GPT-3.5 to fill in the blank regions with novel APs. Optionally, we provide it with

[1]A website that contains articles about how to do various tasks. wikihow.com

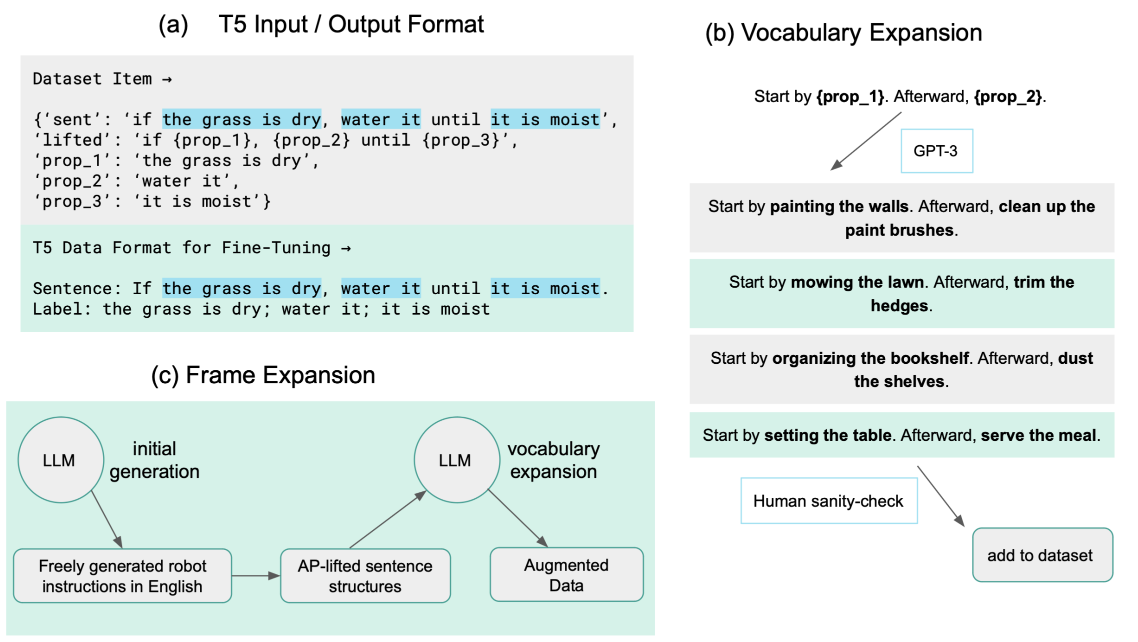
nudges about domains to focus on.

The second aspect of augmentation was **frame expansion**. In this technique, we used GPT-3.5 to generate new sentence structures altogether. We ask for commands that can be provided to a robot, then we process those to get lifted versions and carry out vocabulary expansion on each of the lifted versions. This allows us to get more spontaneous sentence structures in our dataset.

Below is an example of an LLM prompt used for data augmentation through vocabulary expansion. The sections in bold are the nudges meant to balance the domains represented in the dataset, and were frequently changed or removed. The higher-level data augmentation process is illustrated in Figure 4-1.

Example LLM Prompt for Data Augmentation

In the frame given below, fill the props with constraints **that you might find in an airplane or automotive design manual**. You can also fill them with commands that you might give a robot, in a domain such as **navigation, manipulation of objects, or signal detection**. After each sentence, also give me a list of what you used to fill in the props, separated by $$.
Here are some examples: For the frame "If {prop_1}, {prop_2} and {prop_3} until {prop_4}.", an example sentence is "If the trash is full, take it out and dump it until it is emptied." $$ the trash is full $$ take it out $$ dump it $$ it is emptied $$
For the frame "It is equivalent to have {prop_1} and {prop_2} at the same time or to have {prop_3} eventually .", an example sentence is "It is equivalent to have the value of signal 2 be 5.0 and switch 6A to be on at the same time or to have signal 3 become 20.5 eventually." $$ the value of signal 2 be 5.0 $$ switch 6A to be on $$ signal 3 become 20.5 $$
Here is a new frame: {prop_1} until {prop_2} and {prop_3}. Five different example sentences for this can be:



```
Dataset Item →

{'sent': 'if the grass is dry, water it until it is moist',
'lifted': 'if {prop_1}, {prop_2} until {prop_3}',
'prop_1': 'the grass is dry',
'prop_2': 'water it',
'prop_3': 'it is moist'}

T5 Data Format for Fine-Tuning →

Sentence: If the grass is dry, water it until it is moist.
Label: the grass is dry; water it; it is moist
```

Start by **{prop_1}**. Afterward, **{prop_2}**.

Start by **painting the walls**. Afterward, **clean up the paint brushes**.

Start by **mowing the lawn**. Afterward, **trim the hedges**.

Start by **organizing the bookshelf**. Afterward, **dust the shelves**.

Start by **setting the table**. Afterward, **serve the meal**.

Figure 4-1: (a) The format of dataset items and conversion into a text-to-text task. (b) An example of data augmentation through vocabulary expansion. (c) A schematic of data augmentation through frame expansion.

4.1.3 Dataset Characteristics

The complexity and diversity of our dataset were important design considerations, especially since we are working with a small dataset (1.3K items). In this section we analyze our data for complexity and diversity, alongside comparable language-to-TL datasets.

To reflect complexity of instructions, we report the statistics of AP counts per instruction and word counts per instruction in Table 4.1. While the number of APs in the instructions tend to be low, the number of words tends to be high, with sizable standard deviation. It reflects that the AP spans are of variable length.

Table 4.2 shows a comparison of our dataset with other comparable English-to-Formalization datasets, focusing on Temporal Logic datasets. We report a high ratio of unique frames to total sentences, as well as the highest total vocabulary count in spite of having a fairly small dataset.

AP Counts		**Word Counts**	
mean	2.275	mean	11.536
median	2	median	11
max	6	max	34
std. dev.	0.905	std. dev.	4.789

Table 4.1: Statistics of the number of APs and number of total words per instruction in the dataset.

Dataset	**Total Items**	**# Unique Frames**	**Unique : Total Frames**	**Vocab Size**
Our Data	1.3K	193	0.144	**2178**
NL2STL [5]	15K	**14438**	**0.963**	2121
DeepSTL [12]	**120K**	3653	0.030	265
GLTL [10]	11K	193	0.018	193
CW [28]	3.3K	39	0.012	188
Office email [9]	0.15K	23	0.153	143

Table 4.2: Statistics of corpus richness. Compared to previously released large datasets on data diversity metrics [5], our dataset creation method generates a higher ratio of unique frames to total sentences and vocabulary items to total sentences.

4.1.4 Fine-Tuning

Choice of Pretrained Model After preparing the dataset, we fine-tuned a pre-trained language model on the span identification task. We chose the model T5, or Text-to-Text-Transfer-Transformer [24]. T5 is based on the Transformer architecture [29] and treats every task as a text-to-text task. We fine-tuned `t5-small`, the smallest version of T5, in order to place a constraint on model size.

Task Defininition The specific task we are modeling here is identifying spans of the sentence that correspond to potential atomic propositions. Henceforth we will call this 'Span Identification'. The conversion from dataset items to the prompt and label for this task are shown in Figure 4-1. The input is an utterance containing some atomic propositions as well as some operator words and fillers. It is a natural English utterance that might describe a complex task, with temporally interdependent subtasks. The output is a semicolon-separated list of the atomic propositions, in order. Here, the 'atomic propositions' appear as spans of the original text, with no additional modifications. This is done to avoid limiting the system to a particular formal language or a particular representation of an atomic proposition.

4.1.5 Postprocessing

Coreference A potential challenge when breaking a larger instruction into separate spans is the loss of coreference information such as the correct referents of pronouns. To overcome this, we carry out coreference identification on the original instruction. We carry out identification of coreference clusters using an existing external module, `wl-coref` [8]. From there, our system can return coreference-annotated spans or coreference-resolved spans. In the former case, coreference is marked on the entities by marking every entity belonging to the same cluster with the same index. This can be used as a check during grounding to ensure that each entity which shares an index has been grounded to the same real-world object. In the latter case, coreference *resolution*, the antecedent is identified and subsequent spans are edited to include a reference to the main antecedent directly. Addressing coreference modularly with a separate package allows us to incorporate coreference information into our system without complicated re-annotation of our data.

We evaluated this model through multiple accuracy metrics, generalization experiments, and qualitative evaluations to highlight common errors. We also compared performance to few-shot learning using a large language model, here GPT-3. The next two sections focus on the accuracy and generalization evaluations.

4.2 Performance Evaluation

4.2.1 Experimental Methods

Accuracy Metrics

We evaluated accuracy of the T5-based span identification model on four metrics:

- Command accuracy: The percentage of commands for which the full list of APs was perfectly matched with the target

- AP accuracy: The percentage of APs which were perfectly matched with the corresponding targets

- Wrong AP similarity: For all the predicted APs marked as incorrect, their average similarity score with the targets.

- Above-threshold Similarity: The percentage of APs for which the cosine similarity between the target & predicted AP exceeded the threshold of 0.8.

Similarity is on a scale of -1 to 1 and is calculated using the `spacy` library [13] Since exact-match accuracy could be affected by things like verb conjugations, punctuation, and even formatting, we did our best to standardize the output prior to evaluation. However, it is not possible to hard-code all the possible exceptions, and as a result, the similarity-based metrics are more stable.

4.2.2 Evaluation Setup

We evaluated on a test set that was a split of our original data, unseen during training. While evaluating, we split the model's generations into lists of spans. In order to normalize the responses for better exact-match accuracy, punctuation was stripped, unimportant words such as implicit 'you' (in imperatives) were dropped, and all verbs were lemmatized. This is not a part of the post-processing of the model outputs, but only a normalization applied for evaluation purposes.

Hyperparameters For the 'T5-Tuned' span identification model in the following experiments, the model `t5-small` was fine-tuned with maximum sequence length of 256, a learning rate of 3e-4, using the ADAM optimizer with $\epsilon = 1e-8$, and batch size 8 for both training and evaluation. It was trained on a subset of the data, with train/validation/test split sizes of 432/107/179 data points respectively. Training was for 20 epochs and it took less than thirty minutes on an NVIDIA GeForce RTX 2080 GPU. The results below are calculated from a single run of the algorithm.

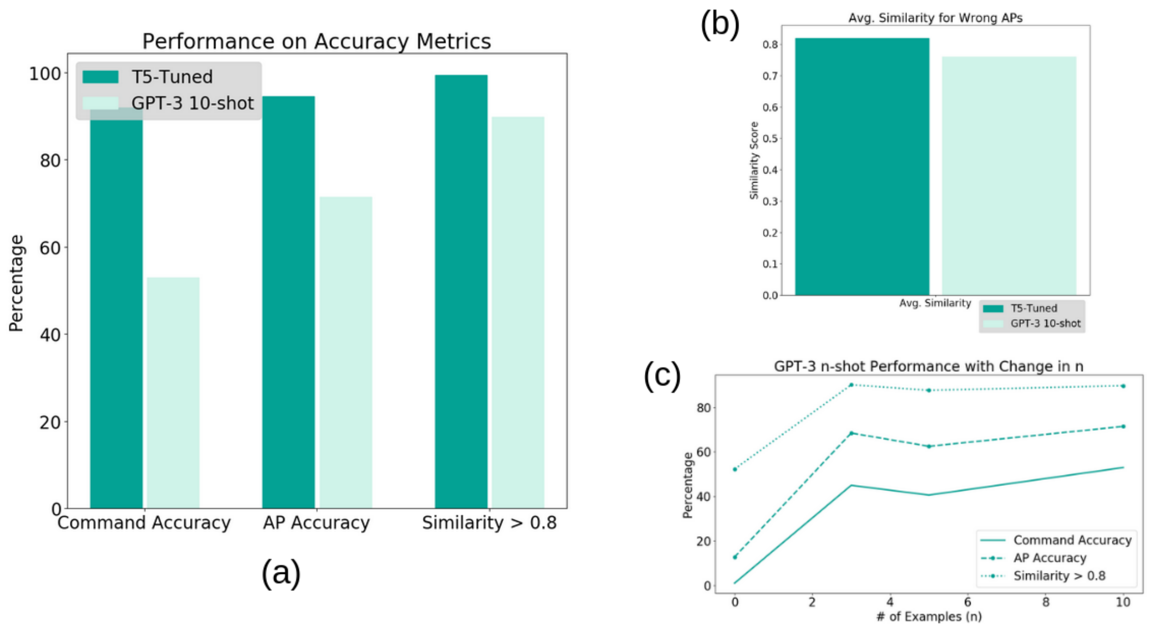


Figure 4-2: **(a)** Performance of the T5-Tuned model on the test set, compared against few-shot performance of GPT-3 on the seed data. **(b)** Performance of the T5-Tuned model and GPT-3 on the 'Wrong AP Similarity' metric. **(c)** Changing the number of examples in GPT-3 few-shot prompting had little effect on accuracy metrics, although it was always better than zero-shot.

4.2.3 Results

The results are shown in Figure 4-2. Our span identification model scores >91% on accuracy metrics. This is compared with few-shot performance of GPT-3 on the seed data. GPT-3 is tested specifically on the natural seed data, not the test data, since the test data may contain examples generated by GPT-3. Numerically, the performance on the accuracy metrics is given in Table 4.3. A qualitative evaluation of the successes and errors is included in the Discussion (Chapter 6).

Metric	T5-Tuned	GPT-3
Command Accuracy	91.95	53.0
AP Accuracy	94.62	71.49
Wrong AP Similarity	0.82	0.76
Similarity > 0.8	99.43	89.79

Table 4.3: Accuracy comparison of two span identification approaches, a fine-tuned `t5-small` model (T5-Tuned) and few-shot GPT-3.

4.3 Cross-Domain Generalization

We tested the ability of the span identification model to generalize to unseen domains and sentence structures. For the experimental setup, we chose five domains with distinct types of instructions: Gardening, Cooking, Cleaning, Navigation, and Manipulation.

4.3.1 Experimental Methods

For each domain, we split our data into a domain-withheld training set that did not contain instructions related to that domain, and a domain-specific test set which contained those instructions. Filtering was based on vocabulary. Common words were identified from each domain, and sentences containing those words were filtered out.

Then, in each case, we freshly fine-tuned a T5-based span identification model on the domain-withheld dataset. We evaluated each model on the domain-specific test set, which contained at least keywords and likely multiple sentence structures that were never encountered during fine-tuning.

4.3.2 Evaluation Setup

The vocabulary used for filtering the datasets for the span ID generalization experiment are given in Table 4.3.2, along with the sizes of each training and testing dataset. The evaluation metrics and T5-Tuning hyperparameters were the same as in section 4.2.

4.3.3 Results

As summarized in Figures 4-3 and 4-4, span identification with a finetuned T5 model shows high generalization potential across domains. Numerical results are shown in Table 4.4. Combined with prior results that show higher generalizability and accuracy for TL translation when an AP-lifted version is used [5], this indicates that models

Task	Words Withheld	Train Size	Test Size
Gardening	plant, soil, water, dig, trim, cut, weed, grass, yard, rake, lawn, wheelbarrow, mow	1240	102
Cleaning	clean, dirty, trash, scrub, dust, sweep, garbage, stain, laundry, dish, wash, organize, tidy	976	366
Manipulation	red, blue, green, purple, arm, robot, box, ball, grab, put, pick, reach, lift, move	732	481
Navigation	turn, walk, go, enter, reach, approach, bedroom, flag, kitchen, navigate, move	751	459
Cooking	stove, cook, pan, stir, boil, cooking, pot, wash, oven, rice, water, batter, meal, dish, sink, spoon, simmer, cut	1085	257

Domain	Command Accuracy (%)	AP Accuracy (%)	Wrong AP Similarity
Gardening	92.1568	95.6175	0.8948
Cooking	91.8287	94.8717	0.8617
Cleaning	93.9890	95.8950	0.8390
Navigation	91.7211	95.1992	0.7494
Manipulation	92.9313	95.9114	0.7703

Table 4.4: Accuracy metrics across domains for the T5-Tuned span identification model. The test domain was withheld during training.

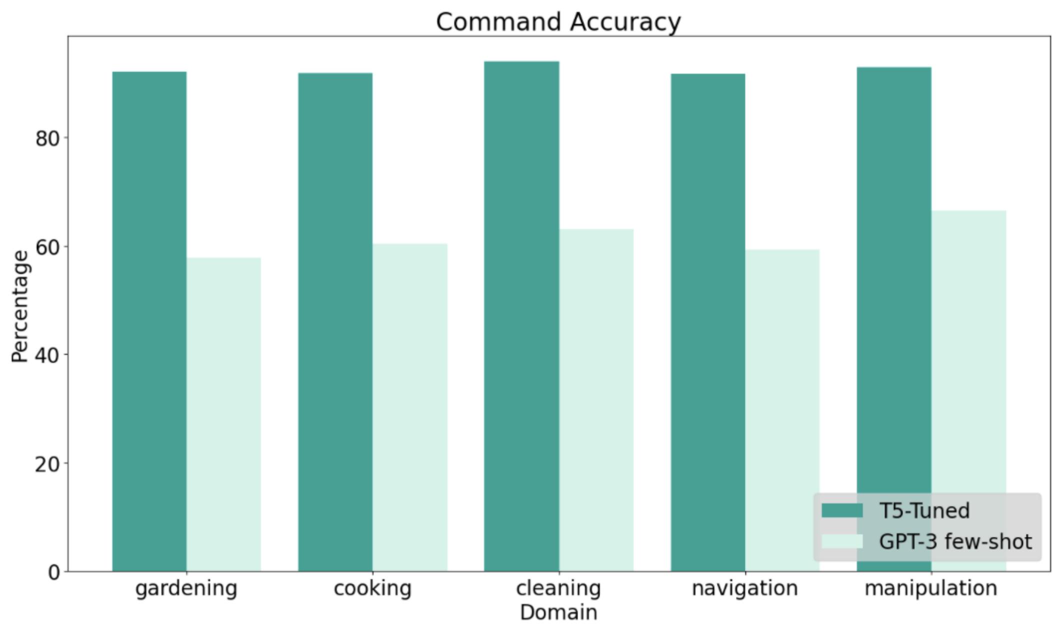
fine-tuned on naturalistic data show good generalization for both parts of the English-to-TL process.

The results for GPT-3 in this experiment are for few-shot prompting, *not* fine-tuning. The purpose of comparing the T5-Tuned model with few-shot GPT-3 is not to contrast the quality of the two pretrained models, but rather the output from two different training approaches – just pre-training on a large amount of data (GPT-3), vs. pre-training a smaller model and fine-tuning it on a small amount of specific data (T5-Tuned). Further, note in this comparison that since our dataset is augmented by GPT-3, many of the test sentences are part of its distribution. Therefore, this experiment does not test the novel domain generalization performance of GPT-3. It visualizes the baseline AP identification performance of GPT-3 alongside the the novel domain generalization performance of the T5-Tuned model. The performance of the T5-Tuned model on **unseen data** is comparable or better than GPT-3 performance on **potentially self-generated data**.

4.4 Conclusions

In this chapter, we implemented and evaluated a model for span identification, the first step towards identifying formalization-agnostic atomic propositions from natural English instructions. We created a novel dataset, which is augmented through an LLM-assisted framework that makes it more natural, complex, and diverse than existing language-to-formalization datasets. After training the span identification model, we evaluated it on four measures of accuracy and generalization to five robot task domains, finding promising cross-domain generalization.

In the next chapter, we discuss methods to convert the output of span identification into atomic propositions given a chosen formalization format as well as experiments on the accuracy and interpretability of such an approach.



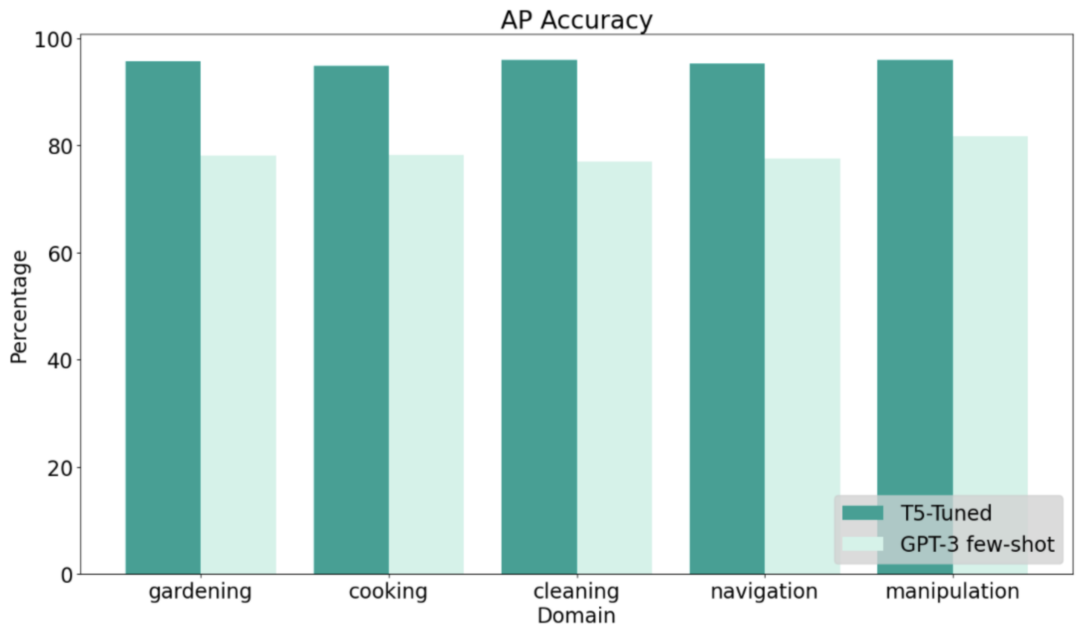
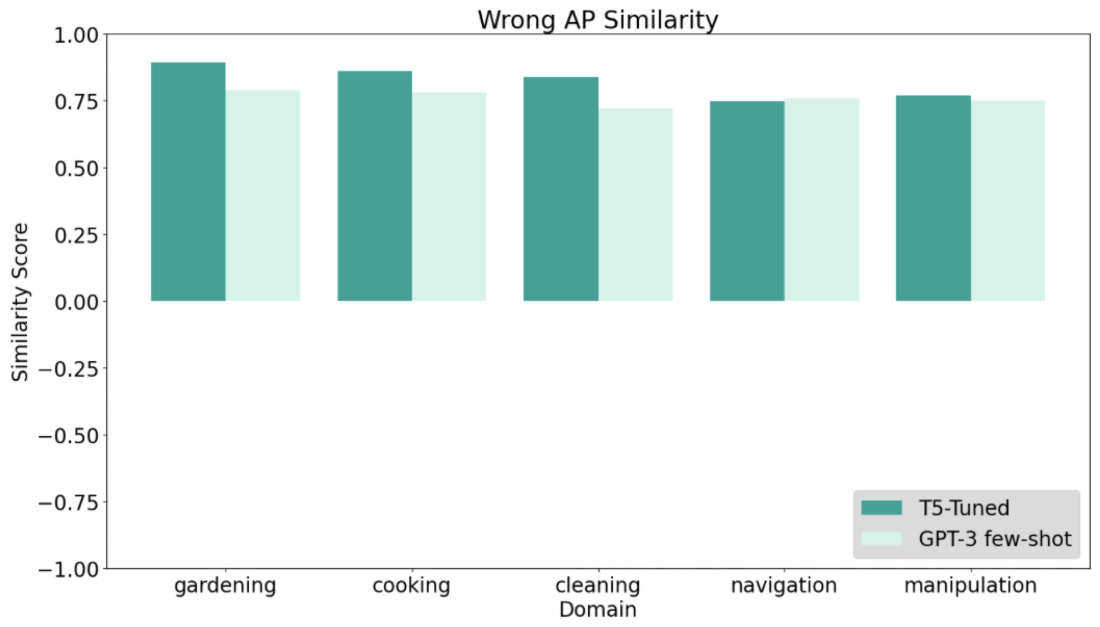


Figure 4-3: Accuracy-based metrics for novel domain generalization performance of T5-Tuned models with certain domains withheld. Compared against baseline performance of 3-shot prompted GPT-3 on the same test sets.



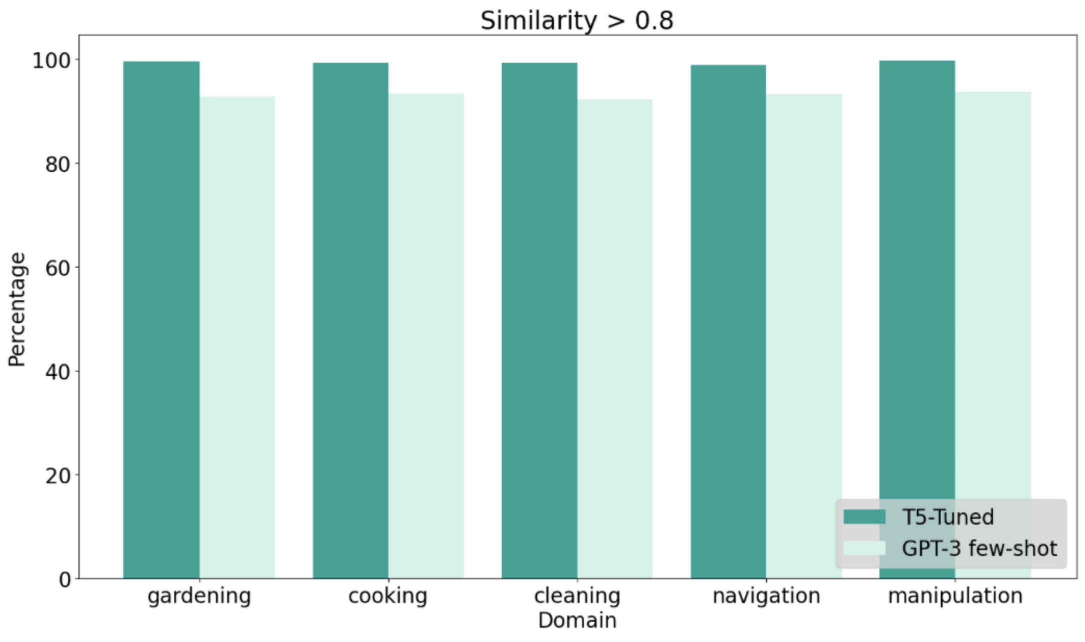


Figure 4-4: Similarity-based metrics for novel domain generalization performance of T5-Tuned models with certain domains withheld. Compared against baseline performance of 3-shot prompted GPT-3 on the same test sets.

Chapter 5

Converting Spans into Atomic Propositions

After identifying spans of the sentence that potentially correspond to atomic propositions, the next step is actually converting these to the representation of APs in the chosen formalization. The method looks different depending on the formalization selected. We have two outputs: the spans, which we have been discussing so far, and the remainder of the sentence ('lifted' sentence), which is presumed to contain the operators. Below, we discuss existing technical approaches for converting the lifted natural language sentence into the skeleton of a logical sentence. Then, we propose a method for converting the identified spans into atomic propositions, in order to fill in the gaps in the logical sentence skeleton.

In order to demonstrate converting the identified spans into atomic propositions, we use two types of representations. One is a custom style where each AP is represented in the format `predicate(object)` using predicates and objects that are existent within the domain. This format can be plugged into a temporal logic skeleton. The second is PDDL, where each AP is represented as a single proposition in the goal condition. These two formalization styles have different syntax, but the approach towards converting the spans into these styles is similar.

After discussing the technical approach to converting identified spans into atomic propositions (either custom style or PDDL goals), we carry out evaluation experiments to test the accuracy, validity, and interpretability of the two-step approach. Sections 5.2 and 5.3 describe test suites designed for the evaluations as well as experimental methods and results. We find that for conversion to Custom-Style APs, the two-step approach improves accuracy and validity of the generated atomic propositions. The results from the PDDL experiment on human-judged accuracy and interpretability are inconclusive, but we present the initial findings and experimental setup and discuss future directions for human user evaluation.

5.1 Technical Approaches

5.1.1 Converting Lifted NL to Temporal Logic

After the spans of interest are extracted, the remaining 'lifted' sentence an be converted to Temporal Logic using existing text-to-text techniques [14, 5]. The APs can be filled back in once the translation is complete. Recent work shows better performance for this 'lifted' translation method than end-to-end translation between English and STL [5]. Using the span identification model completes the pipeline by creating reliable lifted sentences across a variety of domains.

5.1.2 AP Formalizations Using LLM In-Context Learning

We can use in-context learning with a pretrained LLM to match each identified span to one or more APs as defined in the desired definition language. For example, consider a language which has a set of predicates and a set of objects, and the desired representation The prompt contains a domain description in the form of predicates and objects as well as the initial conditions that define the problem. The prompt also contains one unrelated example domain and potential instructions with their ideal responses. APs from the English instruction are plugged into the prompt sequentially. For each one until completion or failure, the model generates either (a) goal conditions, or (b) feedback to the user when an atomic proposition doesn't fit the possibilities of the domain. In our experiments, we find that the two-step method does a better job of keeping the LLM outputs within-domain.

5.2 Matching to a Custom AP Style

5.2.1 Experimental Setup

In this experiment we evaluate the two-step method on identifying atomic propositions of a particular custom style. For instance, the style we chose is `predicate(object)`. For demonstration, we designed a suite of 100 NL instructions with custom AP ground truths. The natural language instructions were sourced from Wikihow as well as the most complex parts of open robotics datasets [18, 2], while ground truths were annotated manually. They covered five domains – traffic, cleanup, gardening, household, and navigation. We implemented a pipeline that would take the identified spans from the first step and use them as input to an LLM for AP generation.

After post-processing to remove irregularities, we evaluate the generated APs on metrics of accuracy and validity. The metric for 'Accuracy' is the percentage of LLM-generated APs which are contained in the ground truth set of APs for the problem. 'Validity' measures whether the predicates and objects referenced by the generated APs are all contained within the domain definitions. We calculate the percentage of valid APs as well as the number of unique out-of-domain predicates and objects generated by each approach.

We tested two approaches – APs as List and APs as Loop. In the List approach, the identified spans are presented as a list in the LLM prompt and it is only prompted once with the entire list. In the Loop approach, the LLM is prompted multiple times, with only one single span per prompt. The APs are expected to be generated one by one in the Loop approach. We tested two LLMs, GPT-3 and GPT-3.5, with both AP-List and AP-Loop. We compared performance with a Full-NL approach, in which the full natural language instruction is given as input to the LLM instead of the identified spans.

Approach	Percentage Accurate	Total APs
AP-List; GPT3.5	58.39	805
AP-Loop; GPT3.5	**64.27**	722
Full-NL; GPT3.5	52.53	1028
AP-List; GPT3	**62.59**	818
AP-Loop; GPT3	61.66	806
Full-NL; GPT3	57.49	1195

Table 5.1: The **accuracy percentage** for four different approaches to formalizing identified spans into a custom style of APs, along with the total APs evaluated.

5.2.2 Results

Accuracy

Results for accuracy are given in Table 5.1. The two-step method works better overall, with the AP-Loop approach on GPT3.5 showing the best result. We also note the total number of generated APs as compared to accuracy percentages: for instance, even though the Full-NL approaches generate a much higher number of APs, their accuracy scores are low, hence the additional generated APs may not be relevant to the problem.

Validity

For validity, the best results were from the Full-NL GPT-3 model. Among AP-based models using the two-step approach, we found that using the AP-List approach with GPT-3 gave the best validity results. The AP-based approaches are all mostly comparable, as shown in Table 5.2.

Approach	Percentage Valid	Total APs
AP-List; GPT3.5	85.52	725
AP-Loop; GPT3.5	86.86	586
Full-NL; GPT3.5	86.58	774
AP-List; GPT3	**88.67**	768
AP-Loop; GPT3	87.05	664
Full-NL; GPT3	**94.23**	948

Table 5.2: The **validity percentage** for four different approaches to formalizing identified spans into a custom style of APs, along with the total APs evaluated.

Approach	ChatGPT	GPT-3
AP-List	58	**14**
AP-Loop	**26**	33
Full-NL	62	26

Table 5.3: The number of unique **out-of-domain** predicates or objects generated by each approach.

Further, we evaluate how much this validity score is affected by the *same* out-of-domain APs being generated vs. *unique* out-of-domain APs being generated. We find that the AP-Loop approach has the best performance (lowest number of unique out-of-domain APs), as shown in Table 5.3. This is an interesting metric, because having fewer unique out-of-domain APs may make it easier to carry out error correction and improve system performance, leading to a large increase in validity percentage with much less effort.

5.3 Translating to PDDL Goals

5.3.1 Experimental Setup

In order to test the two-step method for grounding English instructions to PDDL, we designed a test suite of 40 English-to-PDDL goal generation prompts for an LLM. For half of these, the model was prompted using the original full English instruction ("Full-NL"). In the other half, we first used our span identification model to extract potential APs and then prompted the model using each AP specifically ("AP-Split"). This second approach is similar to the AP-Loop approach discussed in the previous section.

As a pilot test of accuracy and interpretability of model responses, we surveyed a group of users familiar with formal logic, who either had a background in PDDL or were given an introduction to PDDL prior to completing the survey. Each subject filled out a survey with ten model responses from the initial set of 40. The premise of the survey was that the subject was interacting with and judging the performance of a model which grounds natural language instructions to PDDL, and provides feedback if such a grounding is impossible.

Each question started with a plain English instruction. The user attested that they understood the instruction and found it well-formed. In case any user did not find a particular instruction well-formed (5 out of 120 total cases), the user's remaining answers for that question were filtered out in the data processing stage.

After seeing the plain English instruction, the user saw the system's feedback response to the instruction, also in plain English. This response was generated by prompting GPT-3.5 to generate PDDL goals from English. The user was asked to comment on the transparency of the model state. Given the model's feedback response, the user rated how easily and clearly they were able to infer the state of the model (i.e. success or failure at a particular stage). The description of the domain was hidden from the user at this point.

Then, the domain description was shown. The model's *full response* was also shown. The full response consisted of not just the feedback, but also the generated PDDL. The user was asked to judge the model's generated PDDL (if any) for correctness and adherence to the constraints of the domain. This gives us an estimate for accuracy of the generated PDDL.

Each user also rated the ease of answering the accuracy questions on a 5-point scale. This, in addition to the model state transparency question, gives us an estimate for the interpretability of the model's repsonses.

An example of a full stimulus is given in the Appendix. It shows an example instruction and its corresponding domain description, along with the model responses from the AP-Split approach and the Full-NL approach.

5.3.2 Results

Accuracy

For PDDL and feedback generated by both methods, users judged reasonability and adherence to the domains. For the analysis, each response (yes; no; can't say) is converted to a numerical score (1; -1; 0). The responses are then split by question ID

Reasonable		
Input type	Mean Score	Std. Error
AP-Split	-0.2425	± 0.1446
Full-NL	-0.2628	± 0.1450

Table 5.4: Mean scores, across questions, of model responses judged as 'reasonable' for both the AP-Split and Full-NL approaches.

In-Domain		
Input type	Mean Score	Std. Error
AP-Split	0.5651	± 0.1453
Full-NL	0.5168	± 0.1157

Table 5.5: Mean scores, across questions, of model responses judged as 'within domain' for both the AP-Split and Full-NL approaches.

since multiple annotators judged each question. Each question's score is a weighted average of its individual annotator scores, where the weights are their self-reported PDDL comfort on a scale of 1-5. Results are in Tables 5.3.2 and 5.3.2.

In both reasonability and adherence to domain, we observe a slight advantage of the AP-Split approach, but the difference is not significant. It is worth noting that most questions are judged neutral-to-unreasonable and neutral-to-within-domain.

Interpretability

We were interested in transparency of model state as a measure of interpretability. In order to study this, we separated the responses to this question by unique respondents. We calculated the mean rating given by each user under both the Full-NL and AP-Split conditions, and the difference between that and their overall mean rating. The data suggests that on average, AP-Split responses may rated more transparent than the overall mean and Full-NL responses were rated less transparent than the overall mean, but the difference is not significant. The numerical results are given in Table 5.3.2.

In this question as well, the average transparency score by user was fairly high for almost every user. On a scale of 1-7, nearly every user's average response across questions was greater than or equal to 6 points. (Average transparency scores by user: 6.2, 6.2, 6.1, 7, 6, 5.7, 6.6, 6.9, 6, 5.375, 6.142857143, 6.)

Model Transparency		
Input type	Diff. from Overall Mean	Std. Error
AP-Split	0.0624	± 0.0809
Full-NL	-0.0574	± 0.0838

Table 5.6: The average across users of how their mean transparency ratings for either the AP-Split or the Full-NL approaches differed from their overall mean transparency rating.

Users also rated the ease of judging whether the model's response was reasonable and within-domain. There were no significant differences between Full-NL and AP-Split responses.

5.4 Conclusions

In this section, we implemented two ways of formalizing identified spans from a natural English sentence into APs. In the first, we formalized the span into a custom `predicate(object)` AP format. In the second, we converted each span into a PDDL goal. For both, we used large language models with 1-3 examples provided in the prompt.

From the Custom-Style AP generation, which was evaluated algorithmically, the results indicate that the two-step method of splitting up the spans and using them as input is a promising way to address some of the concerns that come up while using LLMs for reasoning tasks – specifically, accuracy of the generations and adherence to the defined domains.

From the PDDL-goal generation, which was evaluated by a small set of human users, results were inconclusive. Generally, user responses showed very little variability at all. We are interested in running this experiment with other language models, questions involving more complex tasks, and users from a wider variety of backgrounds. Thus, we would like to investigate whether this result is robust or is being influenced by factors external to the hypothesis.

Chapter 6

Discussion

In this chapter, we summarize the findings and contributions from previous chapters, along with a discussion of limitations and future directions to this work.

6.1 Span Identification Performance

6.1.1 Qualitative Evaluation

We conducted a qualitative evaluation of span identification performance by studying all the model predictions that had a similarity of less than 0.8 with the target spans.

Key Strengths Both the finetuned `t5-small` and GPT-3 prompting approaches were able to identify generally accurate spans in a sentence, in sequential order. The most interesting result from the evaluation of the span identification model is the high accuracy and similarity scores from the smallest T5 model, fine-tuned on only a few hundred sentences of parallel data. In addition to our model being made open to use, researchers or engineers who wish to train their own span identification model can do so with relatively little annotated seed data and computing power, using the methodology outlined here.

Key Weaknesses The finetuned `t5-small` model struggled with completely different sentence structures, such as complex circuit descriptions from a different dataset

[12] when they were not part of the training set. It's worth pointing out that these sentences were generated by algorithm and not scraped from natural-language sources. For the GPT-3 prompting approach, it was constrained by the particular examples provided in the prompt. The predicted APs often missed important modifying phrases, such as in the AP 'sweep the floor using the broom', it would miss 'using the broom.'

6.1.2 Generalization to Synthetic Data

In our experiments on generalization, we found good generalization to instructions with natural sentence structures even when they were from unseen domains. This is promising. At the same time, we wondered about performance on datasets with highly different sentence structures. With further experimentation, we found that synthetic sentence structures from a circuit-domain dataset [12] were not as well resolved. We believe this is not an inability to generalize to the circuit domain, but rather an unfamiliarity with the highly artificial sentence structures in that dataset. Regardless, such sentence structures might come up in real-world problem definitions, and it may be desirable for a span identification model to deal with them. We found that including just 50 sentences from the circuit dataset (originally 120K sentences in size) into the training data improved performance on the circuit test data from 18% AP accuracy to 58.5% AP accuracy, a gain of 40%. Thus, transfer learning capabilities of T5 can be applied in to achieve large improvements in cross-domain generalization with much less data.

6.2 AP Formalization Performance

For generating custom-styled APs or grounding to PDDL, we proposed using LLMs. There is a tradeoff to consider here. On one hand, including the whole natural language instruction without any breakdown incorporates all the necessary context of the sentence. On the other hand, LLMs are prone to predicting text that is incorrect in a particular context but may have been from an unrelated part of the prompt. By carrying out the span identification chunking, we encourage the grounding done via LLMs to be specific and relevant. This is particularly reflected in the low number of out-of-domain actions and objects generated by the AP-Loop approach, as compared to Full-NL (Table 5.3).

6.3 Contributions in the Context of Contemporary Literature

One line of work in language-based-planning for robotics is grounding natural language instructions directly to the robot's environment or potential plans [21, 27]. Our work is similar to the above in that it focuses on interpreting natural language commands for a robot. It differs from the above by focusing on the language interpretation and not planning, and by making intermediate representations an explicit concern. Choosing the ideal intermediate representation between language and a robot's plan is a difficult question, so we aim to generalize across the choice of formalization.

Parallel data that maps language to formal representations or even atomic propositions is rare, creating a bottleneck for supervised learning. There are various workarounds, largely from the field of language-to-temporal logic. Some work generates artificial data by using rule-based TL-to-language algorithms [12] or soliciting human annotations for simulated robot actions [30]. Our work makes a contribution to this field by creating a dataset and describing the methodology, enabling easier access to parallel training data without the need to create unnatural synthetic sentences.

We use pretrained, Transformer-based [29] neural language models at both stages. With the advent of Large Language Models (LLMs), there is a lot of ongoing research about carrying out tasks end-to-end using a language model. We take an alternative, modular approach with the goal of building a targeted and transparent system.

6.4 Limitations and Future Work

One key limitation is that this work focuses on English data. While cross-lingual transfer learning is an ongoing research problem [3], studies in cross-lingual transfer learning have reported results that are of less quality than monolingual training [26]. As such, the authors think that a language-agnostic approach to the AP identification task would be a valuable contribution. One way to do this could be starting not with the base natural language instruction, but with its dependency tree that is parsed using standard cross-lingual conventions [6].

A second limitation is that we limit this paper to commercially available LLMs (GPT-3 and GPT-3.5). It may be worth evaluating this approach on other LLMs, particularly open-sourced alternatives.

Finally, we carry out a pilot study evaluating the ways in which human users interact with the two different approaches to formalization – our proposed system vs. an end-to-end approach. The users in our study are not necessarily domain experts, but they have a greater exposure to formal representations than the general population. A larger-scale study with the general population would be an interesting future direction of work to study whether this approach makes a difference in the accessibility of formal representations to a non-expert.

6.5 Conclusions

In this thesis, we present a two-step method for identifying atomic propositions from English language instructions. Our method goes from English to span identification and then to the desired representation of atomic propositions, allowing generalization across formalization types. For span identification, we find high cross-domain generalization, suggesting that this may be a promising approach for generalizable AP identification. We also carry out experiments with using span identification as an intermediate for LLM-assisted formalization, finding that this approach increases accuracy and relevance of the generated atomic propositions in a custom representation.

Appendix A

Experimental Materials

A.1 Matching to Custom AP Style

Following is an example of a domain descriptions used for the Custom AP style experiments. There were five domains. We also include an example English language instruction.

Domain Name: Traffic

Predicates:

`decrease_speed, locate_object, not, turn_into, collide, enter, aligned, engage_signal, check_sensor, exists, stop_at, approach, yield`

Objects:

`traffic_sensor, car, traffic, exit, mirror_right, mirror_left, curb, self, traffic_gap, parking_spot, turn_signal, object, parking_line, roundabout`

Example Instruction: Decrease your vehicle's speed as you approach the circular roundabout. Check the left mirror before entering the roundabout and allow any approaching traffic to pass. Enter only when you have a safe gap. If the roundabout is vacant, you can enter without yielding. Turn on your signal as you approach your desired exit.

A.2 PDDL Goal Generation

Below is an example of an instruction and domain description, along with the responses from the AP-split approach and Full-NL-Instruction approach.

Instruction: *Start by vacuuming the rug to remove any visible dust, then fill the bucket with water and mix in a scoop of detergent. Finally, take the rug off the floor and put it in the bucket, please.*

`Domain Description`

`Predicates:`

```
(:predicates
(inside ?obj1 ?obj2)
(nextto ?obj1 ?obj2)
(ontop ?obj1 ?obj2)
(broken ?obj1)
(burnt ?obj1)
(cooked ?obj1)
(dusty ?obj1)
(frozen ?obj1)
(open ?obj1)
(perished ?obj1)
(screwed ?obj1)
(stained ?obj1)
(sliced ?obj1)
(timeset ?obj1)
)
```

`Objects:`

```
(:objects
vacuum.n.04_1 - vacuum.n.04
rug.n.01_1 - rug.n.01
floor.n.01_1 - floor.n.01
agent.n.01_1 - agent.n.01
bucket.n.02_1 - bucket.n.02
scoop.n.02_1 - scoop.n.02
)
```

```
   Initial Conditions:
(:init
(ontop vacuum.n.04_1 rug.n.01_1)
(ontop agent.n.01_1 floor.n.0_1)
(dusty rug.n.01_1)
)
```

Model Response; Full-NL Approach

Attempting to ground ‘Start by vacuuming the rug to remove any visible dust, then fill the bucket with water and mix in a scoop of detergent. Finally, take the rug off the floor and put it in the bucket, please.’...
Feedback: Error, no predicate found for ‘fill’.
[No PDDL generated]

Model Response; AP-Loop Approach

```
Attempting to ground ‘vacuuming the rug.’...
Grounded successfully.
PDDL Goal:
(:goal
(and
(not (dusty rug.n.01_1))
)
)

Attempting to ground ‘remove any visible dust.’...
Grounded successfully.
PDDL Goal:
(:goal
(not (dusty ?obj1))
)

Attempting to ground ‘fill the bucket with water.’...
Grounded successfully.
PDDL Goal:
(:goal
(inside bucket.n.02_1 water)
)

Attempting to ground ‘mix in a scoop of detergent.’...
Feedback: Error, no object found for ‘detergent’. Would you like to add an object called ‘detergent’?
```

Bibliography

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022.

[2] Dilip Arumugam, Siddharth Karamcheti, Nakul Gopalan, Lawson L. S. Wong, and Stefanie Tellex. Accurately and efficiently interpreting human-robot instructions of varying granularities. *CoRR*, abs/1704.06616, 2017.

[3] Mihaela A. Bornea, Lin Pan, Sara Rosenthal, Radu Florian, and Avirup Sil. Multilingual transfer learning for QA using translation as data augmentation. *CoRR*, abs/2012.05958, 2020.

[4] Igor Buzhinsky. Formalization of natural language requirements into temporal logics: a survey. 07 2019.

[5] Yongchao Chen, Rujul Gandhi, Yang Zhang, and Chuchu Fan. Nl2tl: Transforming natural languages to temporal logics using large language models, 2023.

[6] Marie-Catherine de Marneffe, Christopher D. Manning, Joakim Nivre, and Daniel Zeman. Universal Dependencies. *Computational Linguistics*, 47(2):255–308, 07 2021.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[8] Vladimir Dobrovolskii. Word-level coreference resolution. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7670–7675, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[9] 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.

[10] Nakul Gopalan, Dilip Arumugam, Lawson L. S. Wong, and Stefanie Tellex. Sequence-to-sequence language grounding of non-markovian task specifications. In *Robotics: Science and Systems*, 2018.

[11] 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.

[12] Jie He, Ezio Bartocci, Dejan Ničković, Haris Isakovic, and Radu Grosu. Deepstl – from english requirements to signal temporal logic, 2021.

[13] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

[14] Eric Hsiung, Hiloni Mehta, Junchi Chu, Xinyu Liu, Roma Patel, Stefanie Tellex, and George Dimitri Konidaris. Generalizing to new domains by mapping natural language to lifted ltl. *2022 International Conference on Robotics and Automation (ICRA)*, pages 3624–3630, 2022.

[15] Hadas Kress-Gazit, Georgios E. Fainekos, and George J. Pappas. Translating structured english to robot controllers. *Advanced Robotics*, 22(12):1343–1359, 2008.

[16] S. Lauria, T. Kyriacou, G. Bugmann, J. Bos, and E. Klein. Converting natural language route instructions into robot-executable procedures. pages 223–228, 2002.

[17] Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, Andrey Kurenkov, C. Karen Liu, Hyowon Gweon, Jiajun Wu, Li Fei-Fei, and Silvio Savarese. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. *CoRR*, abs/2108.03272, 2021.

[18] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, Mona Anvari, Minjune Hwang, Manasi Sharma, Arman Aydin, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Silvio Savarese,

Hyowon Gweon, Karen Liu, Jiajun Wu, and Li Fei-Fei. BEHAVIOR-1k: A benchmark for embodied AI with 1,000 everyday activities and realistic simulation. In *6th Annual Conference on Robot Learning*, 2022.

[19] Mark Y. Liberman. The trend towards statistical models in natural language processing. 1991.

[20] 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. 1998.

[21] Daniel Nyga, Subhro Roy, Rohan Paul, Daehyung Park, Mihai Pomarlan, Michael Beetz, and Nicholas Roy. Grounding robot plans from natural language instructions with incomplete world knowledge. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 714–723. PMLR, 29–31 Oct 2018.

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

[23] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

[24] 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. *CoRR*, abs/1910.10683, 2019.

[25] Vasumathi Raman, Constantine Lignos, Cameron Finucane, Kenton C. T. 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.

[26] Sebastian Schuster, Sonal Gupta, Rushin Shah, and Mike Lewis. Cross-lingual transfer learning for multilingual task oriented dialog. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3795–3805, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[27] Pararth Shah, Marek Fiser, Aleksandra Faust, J. Chase Kew, and Dilek Hakkani-Tür. Follownet: Robot navigation by following natural language directions with deep reinforcement learning. *CoRR*, abs/1805.06150, 2018.

[28] Shawn Squire, Stefanie Tellex, Dilip Arumugam, and Lei Yang. Grounding english commands to reward functions. In *Robotics: Science and Systems*, 2015.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *ArXiv*, 2017.

[30] 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 Jens Kober, Fabio Ramos, and Claire Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 1706–1718. PMLR, 16–18 Nov 2021.

[31] David HD Warren and Fernando CN Pereira. An efficient easily adaptable system for interpreting natural language queries. *American journal of computational linguistics*, 8(3-4):110–122, 1982.

[32] Terry Winograd et al. Shrdlu: A system for dialog. *Ill and Diagrams Includes Bibliography*, 2:20–48, 1972.

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