# Data-Efficient Learning of Natural Language to Linear Temporal Logic Translators for Robot Task Specification

Jiayi Pan, Glen Chou, and Dmitry Berenson

***Abstract*— To make robots accessible to a broad audience, it is critical to endow them with the ability to take universal modes of communication, like commands given in natural language, and extract a concrete desired task specification, defined using a formal language like linear temporal logic (LTL). In this paper, we present a learning-based approach for translating from natural language commands to LTL specifications with very limited human-labeled training data. This is in stark contrast to existing natural-language to LTL translators, which require large human-labeled datasets, often in the form of labeled pairs of LTL formulas and natural language commands, to train the translator. To reduce reliance on human data, our approach generates a large synthetic training dataset through algorithmic generation of LTL formulas, conversion to structured English, and then exploiting the paraphrasing capabilities of modern large language models (LLMs) to synthesize a diverse corpus of natural language commands corresponding to the LTL formulas. We use this generated data to finetune an LLM and apply a constrained decoding procedure at inference time to ensure the returned LTL formula is syntactically correct. We evaluate our approach on three existing LTL/natural language datasets and show that we can translate natural language commands at 75% accuracy with far less human data ($\leq$12 annotations). Moreover, when training on large human-annotated datasets, our method achieves higher test accuracy (95% on average) than prior work. Finally, we show the translated formulas can be used to plan long-horizon, multi-stage tasks on a 12D quadrotor.**

## I. Introduction

Many tasks that we want our robots to complete are temporally-extended and multi-stage in nature. For example, the success of cooking, urban navigation, robotic assembly, etc. is determined not by a single goal, but rather a sequence of interconnected subtasks and time-varying constraints. Thus, to reliably complete such tasks, it is critical to have an unambiguous specification of these goals and constraints.

Linear temporal logic (LTL) [1] is a powerful and expressive tool for unambiguously specifying temporally-extended tasks. LTL augments the traditional notions of standard propositional logic with temporal operators that are able to express properties holding over trajectories; to complete the task, low-level robot trajectories that satisfy the LTL formula can then be synthesized [2] to complete the task. Despite its strength in specifying complex tasks, LTL is difficult to use for non-expert end users [3], [4], and it is unreasonable to expect an end user to provide an LTL formula that encodes the desired task for many applications. In contrast, it is easy for humans to provide natural language commands.

[1]University of Michigan, Ann Arbor, MI, USA, 48109, {jiayipan, gchou, dmitryb}@umich.edu. This work was supported in part by the Office of Naval Research Grant N00014-21-1-2118 and NSF grants IIS-1750489 and IIS-2113401.

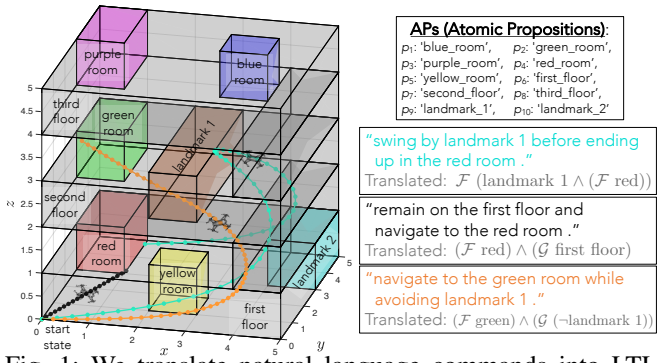


Fig. 1: We translate natural language commands into LTL formulas that achieve complex tasks on a 12D quadrotor.

Thus, a semantic parser which can translate natural language commands into LTL specifications is of great interest.

However, training a task-specific semantic parser can be difficult, and requires a large dataset of natural language commands paired with corresponding LTL formulas [5], [6]; in particular, to use neural architectures, thousands of annotated examples and hundreds of human workers [5], may be required for good generalization. This is prohibitively expensive to collect and is prone to labeling errors, unless LTL experts are used to annotate the data – hence, obtaining data is the key challenge facing LTL translation. In contrast, recent semantic parsing work in the natural language processing (NLP) community [7] has alleviated the need for human-annotated data via synthetic training data [8], [9] and the built-in natural language understanding of pre-trained large language models (LLMs) like GPT-3 [10] and BART [11].

In this paper, we reduce the human-labeled data requirements for natural language-to-LTL translators by applying ideas from low-resource semantic parsing. We assume we are given a predefined set of possible LTL formulas and atomic propositions, and up to one natural language annotation for each formula. We translate these pre-defined formulas to (structured) English either by a rule-based translator when the dataset is sufficiently structured, or by querying a human expert for a translation template, and then using the paraphrasing abilities of modern LLMs [10] to generate a large corpus of diverse natural language commands with similar meaning to the associated LTL formulas. We then use this data to finetune an LLM. Here, we explore two variants, where for training labels we use 1) raw LTL formulas, or 2) a canonical form of the LTL formulas [12]) (an intermediate representation between LTL and English). At evaluation time, we enforce the LLM's output to be syntactically consistent with LTL via constrained decoding. We evaluate our approach on several existing datasets of paired LTL and

natural language commands [6], [5], and show our method achieves competitive performance with prior work (trained on thousands of human annotations), with $\leq$12 human-labeled annotations. Moreover, when combined with human-labeled data, our method exhibits improved generalization compared to prior work. Overall, our contributions are:

- data augmentation schemes for training natural language-to-LTL translators with very few human annotations,
- a neural translation architecture which draws from recent advances in the semantic parsing community to improve LTL translation performance,
- evaluation on several datasets in the literature, achieving competitive performance with far fewer human labels.

## II. Related Work

First, our work is related to methods which aim to obtain task constraints and LTL specifications from human interaction. A large body of work uses interactive training [13], [14] and physical demonstrations to infer task constraints [15], [16], [17], [18] and LTL formulas [19], [20], [21], [22]. However, these forms of human interactions tend to be costly; hence, our goal in this work is to recover LTL formulas from cheaper input, e.g., natural language commands.

If we specify the interaction medium to be language, there is extensive work on translating English to LTL. Early work [23], [24], [25] translated structured English commands to LTL formulas (possibly through an intermediate structured representation); however, providing English commands with this structure requires an understanding of the specific grammar used, which can be unnatural for humans. More recent work uses neural networks to train the translator using thousands of human-labeled natural language/LTL pairs [5], [6], [26]. To reduce the need for human labels, other work aims to learn from trajectories paired with natural language; this however, still requires many trajectories (i.e., demonstrations or executions) to implicitly supervise the translator [27] [28]. Other work [29] [30] improves the translators' generalization to new domains; this is complementary to our method, which improves accuracy within a given set of domains and reduces reliance on human-labeled data. Other work directly translates language to actions [31], [32] without using LTL, and thus cannot use the planning tools [2] that we can exploit.

Our work also relates to the problem of semantic parsing from the NLP community, which seeks to convert from an utterance of (unstructured) natural language to a (structured) logical form which is machine-understandable; e.g., between a command expressed in natural language and an explicit query in a SQL database [33]. Recently, significant progress has been made in *low-resource* semantic parsing. Early works in the area [34], [12] proposed to use a "canonical" natural language form, i.e., an alternate representation of the formal syntax that is closer to English, and which was shown to improve performance on complex tasks. More recent work explores low-resource learning by [8], [7] exploiting automatic training data synthesis using a combination of parsing, templating, paraphrasing, and filtering techniques, or by leveraging large language models (LLMs) [35], [9], such as GPT-3 [10] or BART [11] for their improved performance and generalization capabilities. While low-resource semantic parsing is well-studied in NLP, these advances have yet to transfer to natural language-to-LTL translation, which is itself a semantic parsing problem. A key contribution of our work is to bridge the gap between these two communities. Through extensive experiments, we show that recent ideas in low-resource semantic parsing can notably increase the sample efficiency of traditionally data-hungry LTL translators.

## III. Preliminaries and Problem Statement

We first overview the basics of linear temporal logic (Sec. III-A) and modern generative language models (Sec. III-B), and then give our problem statement (Sec. III-C).

### A. Linear temporal logic (LTL)

We consider planning for discrete-time systems $x_{t+1} = f(x_t, u_t)$, with state $x \in \mathcal{X}$ and control $u \in \mathcal{U}$. To specify tasks for this system, we use linear temporal logic (LTL) [1], which augments standard propositional logic to express properties holding on system trajectories over periods of time. Similar to [5], the LTL specifications considered in this paper can be written with the grammar

$$\varphi ::= p \mid \neg p \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \mathcal{G}\varphi \mid \mathcal{F}\varphi \mid \varphi_1 \,\mathcal{U}\, \varphi_2, \quad (1)$$

where $p \in \mathcal{P} \doteq \{p_i\}_{i=1}^{N_{\text{AP}}}$ are atomic propositions (APs). In this paper, the APs correspond to sets of salient regions of the state space which the robot may wish to visit or avoid (e.g., the blue room in Fig. 1 is an AP). As we consider continuous-state systems in this paper, we associate each AP with a constrained region in the state space; that is, $x \models p_i \Leftrightarrow g_i(x) \leq 0$, for a constraint function $g_i : \mathcal{X} \to \mathbb{R}$. Additionally, $\mathcal{G}\varphi$ denotes that the condition $\varphi$ should hold globally for all time, $\mathcal{F}\varphi$ denotes that $\varphi$ should hold eventually (i.e., there exists some time-step $t$ where $\varphi$ is true), and $\varphi_1 \,\mathcal{U}\, \varphi_2$ denotes that $\varphi_1$ should hold for all time-steps until $\varphi_2$ holds for the first time. This grammar can be used to specify a diverse set of robotic tasks in, e.g., navigation ("drive to the charging station" as $\mathcal{F}\, p_{\text{charging}}$), manipulation ("empty the mug before stacking" as $\neg p_{\text{stack}} \,\mathcal{U}\, p_{\text{empty}}$), etc.

### B. Generative Language Models

Given a piece of text with $n$ words $w_1, w_2, ..., w_n$, a language model will estimate the probability $p(w_1, w_2, ..., w_n)$, for all possible instantiations of text. Auto-regressive language models factor the probability as

$$p(w_1, w_2, ..., w_n) = \prod_{i=1}^{n} p\left(w_i \mid w_1, \ldots, w_{i-1}\right) \quad (2)$$

This formulation of language modeling allows efficient text generation, where given preceding words $w_1, w_2, ..., w_{i-1}$, the model can generate the probability distribution for the next word $p\left(w_i \mid w_1, \ldots, w_{i-1}\right)$.

Modern transformer-based [36] generative language models like GPT-3 [10] and BART [11] can generate text output in an auto-regressive fashion. They are pre-trained on

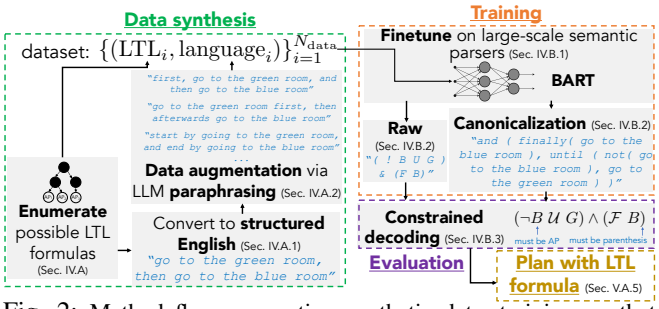


Fig. 2: Method flow: generating synthetic data, training on that data, evaluation, and planning with the evaluated formula.

internet-scale text corpora, and have shown strong natural language understanding and generalization capabilities with impressive performance across many NLP tasks [37].

### C. Problem statement

In this paper, we wish to learn a natural language-to-LTL translator in a data-efficient manner. Specifically, given:

1) a list of possible APs, each with an associated natural language description, e.g., an AP named "G" has the associated description of "inside the green room",
2) a list of possible LTL structures, i.e., a template for an LTL formula with undefined APs, which takes instantiations of those APs as input (this assumption can be relaxed, see Sec. VI),

we wish to learn a mapping between natural language and LTL task specifications, i.e., given a natural language command, we aim to translate it to its associated LTL form.

We consider two data regimes: 1) low-resource scenarios, where we provide limited ($\approx$10) human annotations to train the translator, and 2) the standard data regime, where as in pre-existing language models, we provide thousands of natural language-LTL pairs for training. In low-resource scenarios, we aim to show our method enables satisfactory translation performance, with only a small performance drop relative to a translator trained on a large set of human annotations. In the standard data regime, we aim to show that our translator architecture improves translation accuracy relative to prior methods trained on the same data.

## IV. Method

For data-efficent translation of natural language commands to LTL, our method combines 1) a data synthesis pipeline that automatically generates large synthetic training datasets with little human supervision (Sec. IV-A), and 2) a modern neural semantic parsing architecture that is stronger in natural language understanding and generalization compared to prior work (Sec. IV-B). We visualize our method in Fig. 2.

### A. Data synthesis pipeline

Training a neural translator generally requires a corpus of input and output language pairs, e.g., paired natural language commands and LTL formulas as input and output, respectively. Given the set of possible LTL structures and the set of APs relevant for the set of possible tasks, we can obtain all possible LTL formula outputs by simply filling each LTL structure with combinations of APs. However, while we can generate large numbers of LTL formulas, obtaining a diverse set of natural language descriptions for each LTL formula typically requires a large amount of human labor, making the training extremely expensive [6], [5].

To alleviate this problem, we apply a two-stage pipeline inspired by [8], [7], [12]. First, we perform back-translation (i.e., translate the LTL formula back into *structured* English) and second, perform extensive data augmentation (by leveraging LLMs trained on natural language) to synthesize a diverse set of natural language training data from the LTL formulas, requiring much less human labor. During back-translation, given the LTL formulas used in the task, we generate one natural language description for each LTL formula by using either an LTL-to-English translator (when the LTL representation is sufficiently structured), or templates written by human experts (when such structure does not exist). We discuss specific examples of when to use which in Sec. V. During augmentation, based on the back-translation result, we automatically synthesize a diverse training corpus by leveraging a LLM-based paraphrasing model. We discuss these components in more detail.

*1) Back-translation:* Although mapping natural language into a formal language remains an open research question, the inverse problem of mapping formal language back to natural language can be done relatively easily, by either 1) symbolically parsing the formula [38] or 2) training a neural translator [39]. We build a rule-based LTL-to-English translator based on the grammar of LTL (1). Given an LTL formula, the translator will parse out its syntax tree and then translate it to structured English. This assumed structure renders the translation straightforward. When the LTL corpus is too complex or ambiguous for the translator to work (as in the datasets explored in Sec. V-B and V-C), we obtain the back-translation result by querying human experts to provide a small number of annotations; see Sec. V-B and V-C for specific instances of this process.

*2) Augmentation:* Given the training data obtained in back-translation, unlike previous methods [6], [5], which simply augment the dataset by replacing existing AP combinations with novel ones, we follow [8], [9] and use a neural paraphrasing model to *paraphrase* the text. In particular, we prompt the GPT-3 language model [10] to give ten different paraphrases for every English sentence created during back-translation to augment the synthetic training corpus. An example from the data synthesis pipeline in Sec. V-B is shown below. This example consists of a prompt template (a text template to be filled with string arguments) filled with a source natural language command and then GPT-3's output as the paraphrased results.

> Rephrase the source sentence in 10 different ways.
> Make the outputs as diverse as possible.

Source: Go to the blue room or go to the red room to finally go to the yellow room.

Outputs:

1. You can go to the blue room or the red room, and then finally the yellow room.
2. To get to the yellow room, you must go through the blue room or the red room.
...
10. In order to reach the yellow room, you must first go to the blue room or red room.

Since the back-translated structured English commands are empirically similar to the natural language that GPT-3 is trained on, we find GPT-3 returns meaningful, diverse paraphrases resembling natural language. In short, our insight is to exploit LLMs' large-scale pre-training on general-purpose natural language to generate diverse English commands that notably reduces reliance on human annotators (who may also make mistakes due to unfamiliarity with LTL, cf. Sec. I).

### B. Architecture

Applying large language models to low-resource semantic parsing has led to much progress (see Sec. II). Following [35], we select the pre-trained BART-large language model as our translation model, finetune it on the task-specific corpus (using either raw LTL or a canonicalization of LTL for training labels), and at inference time perform LTL grammar-constrained decoding. We discuss these now in detail.

*1) Pre-training and fine-tuning:* BART is a transformer-based [36] language model. Given a corrupted version of English text as input, e.g., "my is Alex", the model is trained to recover and output the original text 'my name is Alex". In our context, BART is given the natural language command as input, and the LTL formula as output. We explore two variants on the training label representation: 1) using the raw LTL formula for training labels (i.e., $\mathcal{F}\ B$, for "eventually visit the blue room", is transcribed as "F B" for the training label), and 2) using a canonicalization of the LTL formula for the labels (an intermediary between LTL and English), which we describe in Sec. IV-B.2. It is worth noting that our proposed method can be easily applied to other potentially stronger language models like T5-XXL [40] or GPT-3 [10]; we choose BART-large because it has a moderate number of 406M parameters and is efficient to finetune on a single GPU. We use the hyper-parameters from [35] for finetuning.

*2) Canonical form for LTL:* While exploiting the structure in pre-trained LLMs can be fruitful, directly applying them on LTL formulas (especially) can degrade performance. As language models (including BART) are primarily trained on natural language, there is a distribution shift when evaluating on the text transcription of LTL formulas, e.g., "F B" does not resemble natural language. In [11], it was shown that creating a one-to-one mapping from a formal language to a "canonical" representation, which is "closer" to natural language than the raw LTL formula, can mitigate the distribution shift and enable stronger benefits from pre-trained LLMs.

We now describe the canonical form for LTL that we use. Given an LTL formula, we build its equivalent parse tree form (see Fig. 3, and [21] for details), replace the elements

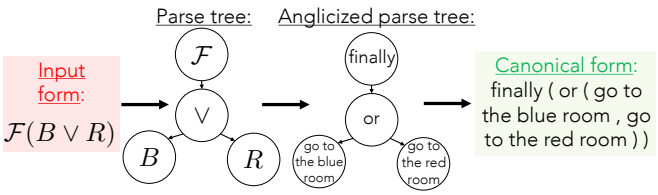


Fig. 3: Transforming from raw LTL to a canonical form.

of the LTL grammar with corresponding English phrases, and starting from the parse tree's root, we transcribe it to text, with parentheses and commas to encompass and separate an operator's input arguments. For example (Fig. 3), consider the formula $\mathcal{F}(B \vee R)$; this can be written as the parse tree in Fig. 3, and after Anglicization and transcription, we have "finally ( or ( go to the blue room , go to the red room ) )".

However, canonicalization also has drawbacks, e.g., 1) it increases the transcription length, which can hurt accuracy, and 2) for simpler tasks, the inductive bias provided by the canonical form may not help as much. Thus, we evaluate both raw and canonicalized LTL in the results to explore which representation is better suited for LTL translation.

*3) Constrained decoding for the language model:* Constrained decoding [35] is a common technique used together with LLMs in low-resource semantic parsing to guarantee that the output will be well-formed. Given a pre-defined set of possible outputs, the system will constrain the LM by only considering the next-token prediction that is in the output sets. In practice, we incorporate the constrained decoding implementation in [35] and provide it with the set of possible LTL formulas in the task obtained in IV-A.

To recap, we synthesize a dataset of natural language/LTL pairs by generating possible LTL formulas, converting them to structured English, and then using LLM paraphrasers to get synthetic natural language commands. This data (either in raw or canonical form) is used to finetune BART, and at evaluation time, we use an LTL-constrained decoder.

## V. Results

To evaluate our approach, we compare our method (the raw LTL and canonical variants are denoted as BART-FT-Raw and BART-FT-Canonical in in Tab. I) with two existing baselines for natural language-to-LTL translation: CopyNet [29], and an RNN with attention mechanism[1] (denoted RNN) [6]. We also examine several ablations of our method, to evaluate the necessity of various components of our pipeline. In particular, we 1) remove constrained decoding at evaluation time, denoted "-NoConstrainedDecoding", and 2) train BART directly on structured English, without paraphrasing (cf. Sec. IV-A.2), denoted "no augmentation" in Tab. I.

We evaluate our method on three datasets of paired LTL formulas and natural language commands: a drone planning dataset (Sec. V-A) [5], an robot navigation dataset (Sec. V-B) [6], and a robot manipulation dataset (Sec. V-C) [6]. We show that 1) despite our limited human-labeled data, we achieve competitive English to LTL translation accuracy

[1]Two variants of RNN models are discussed in [6], which have very similar performance. We select the RNN + Bahdanau Attention architecture [41] for our experiments as it has overall better performance.

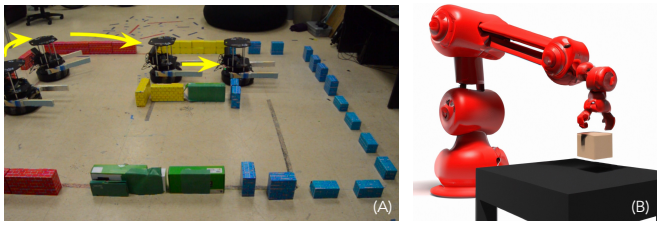


Fig. 4: The evaluation datasets. See Fig. 1 for the drone dataset. (A) Cleanup World [6]. (B) Pick-and-place [6].

on these datasets, and 2) when trained on the datasets, our architecture yields better accuracy than the baselines. Our code is at github.com/UM-ARM-Lab/Efficient-Eng-2-LTL.

### A. Drone planning

*1) Definition:* In this dataset (from [5]), as illustrated in Fig. 1, the task is to translate a natural language command for drone navigation into an LTL expression, which can then be fed into a trajectory planner that completes the task in a pre-defined environment (i.e., if the correspondence between an AP and its real-world region is known). This dataset contains 5 unique LTL structures and 12 different APs, with a total of 6,185 commands for 343 different LTL formulas.

*2) Experimental setup:* To explain the structure and our processing of this dataset, we present an example below. In black, we show the instruction in natural language, followed by the canonical form (see Sec. IV-B.2) used by our method, and the raw LTL representation used by the baseline:

head to the yellow room , but make sure to go through the blue room first .

finally ( and ( the blue room , finally ( the yellow room ) )

F ( blue_room & F ( yellow_room ) )

Of the three considered datasets, the LTL formulas in this dataset are neither too ambiguous nor too complex (see Sec. V-B and V-C respectively for cases where it does not hold) to stop back-translation from functioning. Thus, we first map each original LTL representation to its canonical form via parse tree (cf. Sec. IV-B.2), and then do the back-translation.

*3) Results:* Our translation accuracy on this dataset is presented in Tab. I. The translation output is considered accurate if it matches exactly with the ground truth output. This may be conservative, since some clauses in a formula can be reordered (thus failing to match the output exactly) while retaining identical semantic meaning (see Sec. VI for more discussion). In the training data column of Tab. I, "golden" refers to the human-annotated data from the original drone planning dataset, while "synthetic" refers to the synthetic training data that we obtained by the data synthesis pipeline of Sec. IV-A. As there is no official division of the training / evaluation split when evaluating on the golden dataset, we report accuracy by its five-fold cross-validation result. We generate 5900 synthetic data points and as no golden data is provided to the model for training, we evaluate the model's performance on the full golden dataset.

*4) Discussion:* When using the golden dataset to train, our model performs the best compared to the baseline models, outperforming them in translation accuracy by about 2%. Moreover, our "-Raw" and "-Canonical" variants have similar accuracy. This suggests our architecture has better generalization to unseen data, which can be attributed to 1) our model's higher capacity relative to the baselines, and 2) the extensive pre-training provided by BART (in contrast, only the word embedding layer in the baselines is pre-trained). When we consider the low-resource scenario, our method achieves an accuracy of 69%. Note that 1) reduced accuracy compared to training on the golden dataset is expected, due to the distribution shift between the two datasets, and 2) while application-dependent, accuracies of 70% are common for the state-of-the-art in semantic parsing, e.g., [35]. In contrast, all the baseline methods perform much worse (20-30 %). The ablation of our method without data augmentation does similarly poorly (20-30%), whereas removing constrained decoding causes a slight degradation of 1%. Here, canonicalization hurts performance by 1%; this may be due to the reasons discussed in Sec. IV-B.2. On this dataset, we posit that the combination of the pre-trained LLM, the data augmentation, and constrained decoding enables our accuracy, while canonicalization is not needed.

*5) From LTL formulas to trajectories:* To show that our translated LTL formulas can specify the complex behavior requested in natural language, we compute plans satisfying translated formulas on a quadrotor. It is modeled as a 12D double integrator, where the state is the 3D pose (6 states) and the linear/angular velocity (6 states); we assume we control the accelerations. These are linear dynamics, so dynamically-feasible trajectories satisfying the LTL formulas can be computed with mixed integer convex programming [21], [2]. In Fig. 1, we visualize three plans which satisfy the translated formulas. Here, APs are modeled as polytopes, i.e., $p_i \Leftrightarrow \{x \mid A_i x \leq b_i\}$. Complex behavior arises from the plans, e.g., for the command "swing by landmark 1 before ending up in the red room", the drone visits the second floor without exiting the map (gray), touches landmark 1, and then smoothly returns to the first floor to visit the red room.

### B. Cleanup World

*1) Definition:* The Cleanup World environment [42] (Fig. 4(A)) involves a robot interacting with its environment by moving through different rooms, or by moving objects from one room to another. Based on this environment, [6] collects 3,382 natural language command-LTL pairs, containing 39 LTL formulas with 4 unique LTL structures and 6 unique APs. The task for this domain is to give a natural language command to a simulated robot, which asks it to move through different rooms or asks to move objects to other rooms.

*2) Experiment setup:* As done for the drone dataset, we will present an example of the structure and our processing of this dataset. The color-coding is the same as in Sec. V-A, but this time we show two instructions in natural language that correspond to the same LTL formula.

enter the red room and bring the chair back to the blue room

move into the red room and push the chair back into the purple room

finally ( and ( go to the red room , finally ( go to the blue room with chair ) ) )

F & R F X

TABLE I: Translation accuracy. Ours, baselines, ablations. Top: regular data regime; bottom: low-resource regime. (Number of LTL structures/formulas).

| Model architecture | Training data | Test data | Drone (5/343) | Cleanup (4/39) | Pick (1/5) |
|---|---|---|---|---|---|
| RNN [6] | 4/5 golden | 1/5 golden | 87.18 | 95.51 | 93.78 |
| CopyNet [29] | 4/5 golden | 1/5 golden | 88.97 | 95.47 | 93.14 |
| BART-FT-Raw (ours) | 4/5 golden | 1/5 golden | **90.78** | **97.84** | **95.97** |
| BART-FT-Canonical (ours) | 4/5 golden | 1/5 golden | 90.56 | 97.81 | 95.70 |
| RNN [6] | synthetic | full golden | 22.41 | 52.54 | 32.39 |
| CopyNet [29] | synthetic | full golden | 36.41 | 53.40 | 40.36 |
| BART-FT-Raw (ours) | synthetic | full golden | **69.39** | **78.00** | **81.45** |
| BART-FT-Canonical (ours) | synthetic | full golden | 68.99 | 77.90 | 78.23 |
| BART-FT-Raw-NoConstrainedDecoding | synthetic | full golden | 68.23 | 76.26 | 81.05 |
| BART-FT-Canonical-NoConstrainedDecoding | synthetic | full golden | 67.45 | 72.06 | 69.49 |
| BART-FT-Raw (ours) | synthetic; no augmentation | full golden | 29.43 | 52.51 | 80.38 |
| BART-FT-Canonical (ours) | synthetic; no augmentation | full golden | 39.21 | 53.16 | 67.88 |

This dataset lacks documentation for some APs, i.e., it is unclear what "X" corresponds to in English; without this information, back-translation to structured English cannot be done via our rule-based translator. Moreover, the dataset is highly noisy, e.g., in the second natural language command, the annotator misjudged the color as purple. To handle these challenges, we manually inspect the dataset, and provide the data needed to pair every LTL formula in this domain to a corresponding canonical form/natural language description. Naïvely, this requires 39 annotations (one for each LTL formula in the dataset), but we reduce this to 10 annotations by exploiting the compositional structure of LTL. Specifically, we collect one natural language description for each of the six APs, and the canonical form/natural language description for each of the four LTL structures. It is worth discussing the comparability of data collection costs. Providing a natural language description for the four LTL *structures* may require the human annotator to be more familiar with LTL, while annotating LTL formulas case by case may be easier (has been done with crowd-sourcing [6], though accuracy is still a challenge). Since our pipeline is flexible, one can choose between 10 natural language annotations on LTL structures (more expensive) or 39 cheaper annotations of LTL formulas.

*3) Results and Discussion:* We report our accuracy in Tab. I. The evaluation criteria (exact matching) is the same. When using the golden dataset to train the model, like before, both the raw and canonical variants of our method outperform the baselines by 2%. In the low-resource scenario, we generate 594 synthetic data points, and our method achieves ≈78% accuracy (for both raw and canonical); this is higher than the drone example, and is a ≈20% drop from training on the golden dataset (expected due to distribution shift). In contrast, all baselines perform much worse (≈50%). The ablations also degrade (≈74% when removing constrained decoding, ≈53% when removing augmentation), and the ablated raw and canonical variants perform similarly. Overall, this corroborates the conclusions of Sec. V-A.

### C. Pick-and-place

*1) Definition:* In this dataset [6] (see Fig. 4(B)), the robot conducts repetitive actions based on a user command specified in natural language. It has 5 different LTL formulas with 5 different APs and 1 unique LTL structure.

*2) Experiment setup:* As done for the previous datasets, we will present an example of the structure and our processing of this dataset; color-coding is as before.

scan the empty area of the table and pick up any non green objects moving them to the basket

globally ( and ( until ( scan , not ( any non green cubes ) ) , finally ( any non green cubes ) ) )

G & U S ! C F C

As the LTL structure has a parse tree of depth 5 (i.e., the task is complex), it would require extensive engineering to design the LTL-to-English translator. Thus, we follow the process in Sec. V-B.2, and manually inspect the dataset, giving a total of 5 canonical form/natural language annotations.

*3) Results and Discussion:* Our accuracy on the pick-and-place dataset is presented in Tab. I. The evaluation criteria (exact matching) is the same as before. Again, when training on the golden dataset, our model (both variants) outperforms the baselines by ≈2%. For the low-resource scenario, we generate 55 synthetic data points, and our method (raw) gives an accuracy of 81%; this is comparable with Sec. V-B, and is around a 14% drop from training on the golden dataset, which is a slightly smaller drop compared to the other two datasets. Here, canonicalization hurts accuracy by 3%; this is consistent with Sec. V-A. In contrast, all the baselines perform much worse (32 and 40 %). The ablations of our method also worsen (≈80% for the raw variants and ≈68% for the canonical variants). Surprisingly, "-Raw" degrades less than "-Canonical" (drop of 1 vs. 13%). This may be since: 1) there is only one LTL structure, so only the APs need to be correctly translated for overall correctness, and 2) raw LTL is more compact than the canonical form, so there are fewer words to distract the model in identifying the APs.

Overall, these results are as expected. However, we did not expect "-Raw" to consistently outperform "-Canonical", in contrast to established results, e.g., [34], [12], [35]. We believe that the most likely reason for this (see Sec. IV-B.2 for other ideas) is that our evaluation datasets are not complex enough to benefit from canonicalization. This is consistent with how the accuracy gap for pick-and-place is smaller than the gap for e.g., the more complex drone dataset.

## VI. Discussion and Conclusion

In this paper, we present an approach for translating natural language commands into corresponding LTL formulas. Our method is highly data-efficient, and can achieve 75% translation accuracy with only a handful of ($\leq$ 12) human annotations. We achieve this efficiency through data augmentation and by using this data to finetune an LLM.

Our work has limitations that are interesting directions for future work. First, exploiting the language models' uncer-

tainty (e.g., the top $k$ best formulas) by grounding them to the environment may improve accuracy. Second, we assume a natural language command maps to one LTL formula; however, many natural language commands are inherently ambiguous. Thus, we will study uncertainty-aware planning (e.g., [43], [44]) at the task level, with uncertainty driven by natural language. Third, we assumed we know all possible LTL structures; we will explore automatic synthesis of LTL structures to improve accuracy on unseen LTL structures.

## REFERENCES


[1] C. Baier and J. Katoen, *Principles of model checking*. MIT Press, 2008.

[2] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *53rd IEEE Conference on Decision and Control, CDC 2014, Los Angeles, CA, USA, December 15-17, 2014*. IEEE, 2014, pp. 81–87.

[3] A. Pakonen, C. Pang, I. Buzhinsky, and V. Vyatkin, "User-friendly formal specification languages - conclusions drawn from industrial experience on model checking," in *21st IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2016, Berlin, Germany, September 6-9, 2016*. IEEE, 2016, pp. 1–8.

[4] R. Schlör, B. Josko, and D. Werth, "Using a visual formalism for design verification in industrial environments," in *Services and Visualization: Towards User-Friendly Design, ACoS'98, VISUAL'98, AIN'97, Selected Papers*, ser. Lecture Notes in Computer Science, T. Margaria, B. Steffen, R. Rückert, and J. Posegga, Eds., vol. 1385. Springer, 1998, pp. 208–221.

[5] Y. Oh, R. Patel, T. Nguyen, B. Huang, E. Pavlick, and S. Tellex, "Planning with state abstractions for non-markovian task specifications," in *Robotics: Science and Systems XV, University of Freiburg, Freiburg im Breisgau, Germany, June 22-26, 2019*, A. Bicchi, H. Kress-Gazit, and S. Hutchinson, Eds., 2019.

[6] N. Gopalan, D. Arumugam, L. Wong, and S. Tellex, "Sequence-to-Sequence Language Grounding of Non-Markovian Task Specifications," in *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, Jun. 2018.

[7] P. Yin, J. Wieting, A. Sil, and G. Neubig, "On the ingredients of an effective zero-shot semantic parser," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, S. Muresan, P. Nakov, and A. Villavicencio, Eds. Association for Computational Linguistics, 2022, pp. 1455–1474.

[8] S. Xu, S. Semnani, G. Campagna, and M. Lam, "AutoQA: From Databases To QA Semantic Parsers With Only Synthetic Training Data," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 422–434. [Online]. Available: https://aclanthology.org/2020.emnlp-main.31

[9] S. Rongali, K. Arkoudas, M. Rubino, and W. Hamza, "Training Naturalized Semantic Parsers with Very Little Data," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*. Vienna, Austria: International Joint Conferences on Artificial Intelligence Organization, Jul. 2022, pp. 4353–4359. [Online]. Available: https://www.ijcai.org/proceedings/2022/604

[10] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[11] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, Eds. Association for Computational Linguistics, 2020, pp. 7871–7880.

[12] Y. Wang, J. Berant, and P. Liang, "Building a Semantic Parser Overnight," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 1332–1342. [Online]. Available: https://aclanthology.org/P15-1129

[13] G. Wang, C. Trimbach, J. K. Lee, M. K. Ho, and M. L. Littman, "Teaching a Robot Tasks of Arbitrary Complexity via Human Feedback," in *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*. Cambridge United Kingdom: ACM, Mar. 2020, pp. 649–657.

[14] A. Shah and J. Shah, "Interactive robot training for non-markov tasks," *CoRR*, vol. abs/2003.02232, 2020. [Online]. Available: https://arxiv.org/abs/2003.02232

[15] G. Chou, D. Berenson, and N. Ozay, "Learning constraints from demonstrations with grid and parametric representations," *Int. J. Robotics Res.*, vol. 40, no. 10-11, 2021.

[16] G. Chou, N. Ozay, and D. Berenson, "Learning constraints from locally-optimal demonstrations under cost function uncertainty," *IEEE Robotics Autom. Lett.*, vol. 5, no. 2, pp. 3682–3690, 2020.

[17] C. Pérez-D'Arpino and J. A. Shah, "C-LEARN: learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy," in *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*. IEEE, 2017, pp. 4058–4065.

[18] D. R. R. Scobee and S. S. Sastry, "Maximum likelihood constraint inference for inverse reinforcement learning," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[19] G. Chou, N. Ozay, and D. Berenson, "Learning temporal logic formulas from suboptimal demonstrations: theory and experiments," *Autonomous Robots*, vol. 46, no. 1, pp. 149–174, Jan. 2022. [Online]. Available: https://doi.org/10.1007/s10514-021-10004-x

[20] A. Shah, P. Kamath, J. A. Shah, and S. Li, "Bayesian inference of temporal task specifications from demonstrations," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 2018, pp. 3808–3817.

[21] G. Chou, N. Ozay, and D. Berenson, "Explaining Multi-stage Tasks by Learning Temporal Logic Formulas from Suboptimal Demonstrations," arXiv, Tech. Rep. arXiv:2006.02411, Jun. 2020, arXiv:2006.02411 [cs, eess] type: article. [Online]. Available: http://arxiv.org/abs/2006.02411

[22] M. Vazquez-Chanlatte, S. Jha, A. Tiwari, M. K. Ho, and S. A. Seshia, "Learning task specifications from demonstrations," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 2018, pp. 5372–5382.

[23] C. Finucane, G. Jing, and H. Kress-Gazit, "LTLMoP: Experimenting with language, Temporal Logic and robot control," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2010, pp. 1988–1993, iSSN: 2153-0866.

[24] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Translating Structured English to Robot Controllers," *Advanced Robotics*, vol. 22, no. 12, pp. 1343–1359, Jan. 2008.

[25] A. P. Nikora and G. Balcom, *Automated Identification of LTL Patterns in Natural Language Requirements*, 2009.

[26] C. Hahn, F. Schmitt, J. J. Tillman, N. Metzger, J. Siber, and B. Finkbeiner, "Formal Specifications from Natural Language," Jun. 2022, arXiv:2206.01962 [cs]. [Online]. Available: http://arxiv.org/abs/2206.01962

[27] R. Patel, E. Pavlick, and S. Tellex, "Grounding language to non-markovian tasks with no supervision of task specifications," in *Robotics: Science and Systems XVI, Virtual Event / Corvalis, Oregon, USA, July 12-16, 2020*, 2020.

[28] C. Wang, C. Ross, Y. Kuo, B. Katz, and A. Barbu, "Learning a natural-language to LTL executable semantic parser for grounded robotics," in *4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA*, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. J. Tomlin, Eds., vol. 155. PMLR, 2020, pp. 1706–1718.

[29] M. Berg, D. Bayazit, R. Mathew, A. Rotter-Aboyoun, E. Pavlick, and S. Tellex, "Grounding language to landmarks in arbitrary outdoor environments," in *2020 IEEE International Conference on Robotics*

*and Automation, ICRA 2020, Paris, France, May 31 - August 31, 2020*. IEEE, 2020, pp. 208–215.

[30] E. Hsiung, H. Mehta, J. Chu, J. X. Liu, R. Patel, S. Tellex, and G. Konidaris, "Generalizing to new domains by mapping natural language to lifted LTL," in *2022 International Conference on Robotics and Automation, ICRA 2022, Philadelphia, PA, USA, May 23-27, 2022*. IEEE, 2022, pp. 3624–3630.

[31] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra, "Embodied question answering," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 1–10.

[32] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. D. Reid, S. Gould, and A. van den Hengel, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 3674–3683.

[33] Y. Sun, D. Tang, N. Duan, J. Ji, G. Cao, X. Feng, B. Qin, T. Liu, and M. Zhou, "Semantic parsing with syntax- and table-aware SQL generation," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 361–372. [Online]. Available: https://aclanthology.org/P18-1034

[34] J. Berant and P. Liang, "Semantic parsing via paraphrasing," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 1415–1425. [Online]. Available: https://aclanthology.org/P14-1133

[35] R. Shin, C. H. Lin, S. Thomson, C. Chen, S. Roy, E. A. Platanios, A. Pauls, D. Klein, J. Eisner, and B. Van Durme, "Constrained Language Models Yield Few-Shot Semantic Parsers," Nov. 2021, arXiv:2104.08768 [cs]. [Online]. Available: http://arxiv.org/abs/2104.08768

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[37] A. Srivastava, A. Rastogi, A. Rao, A. A. M. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso *et al.*, "Beyond the imitation game: Quantifying and extrapolating the capabilities of language models," *arXiv preprint arXiv:2206.04615*, 2022.

[38] A. Ranta, "Translating between language and logic: what is easy and what is difficult," in *International Conference on Automated Deduction*. Springer, 2011, pp. 5–25.

[39] H. Cherukuri, A. Ferrari, and P. Spoletini, "Towards explainable formal methods: From ltl to natural language with neural machine translation," in *Requirements Engineering: Foundation for Software Quality: 28th International Working Conference, REFSQ 2022, Birmingham, UK, March 21–24, 2022, Proceedings*. Berlin, Heidelberg: Springer-Verlag, 2022, p. 79–86. [Online]. Available: https://doi.org/10.1007/978-3-030-98464-9_7

[40] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020. [Online]. Available: http://jmlr.org/papers/v21/20-074.html

[41] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2015.

[42] J. MacGlashan, M. Babes-Vroman, M. desJardins, M. L. Littman, S. Muresan, S. Squire, S. Tellex, D. Arumugam, and L. Yang, "Grounding english commands to reward functions," in *Robotics: Science and Systems*, 2015.

[43] G. Chou, D. Berenson, and N. Ozay, "Uncertainty-aware constraint learning for adaptive safe motion planning from demonstrations," in *4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA*, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. J. Tomlin, Eds., vol. 155. PMLR, 2020, pp. 1612–1639.

[44] G. Chou, H. Wang, and D. Berenson, "Gaussian process constraint learning for scalable chance-constrained motion planning from demonstrations," *IEEE Robotics Autom. Lett.*, vol. 7, no. 2, pp. 3827–3834, 2022.