On syntax and semantics for voice assistants in autonomous vehicles

Warrick Scott Macmillan
Developed with the support of Ekaterina Komendantskaya

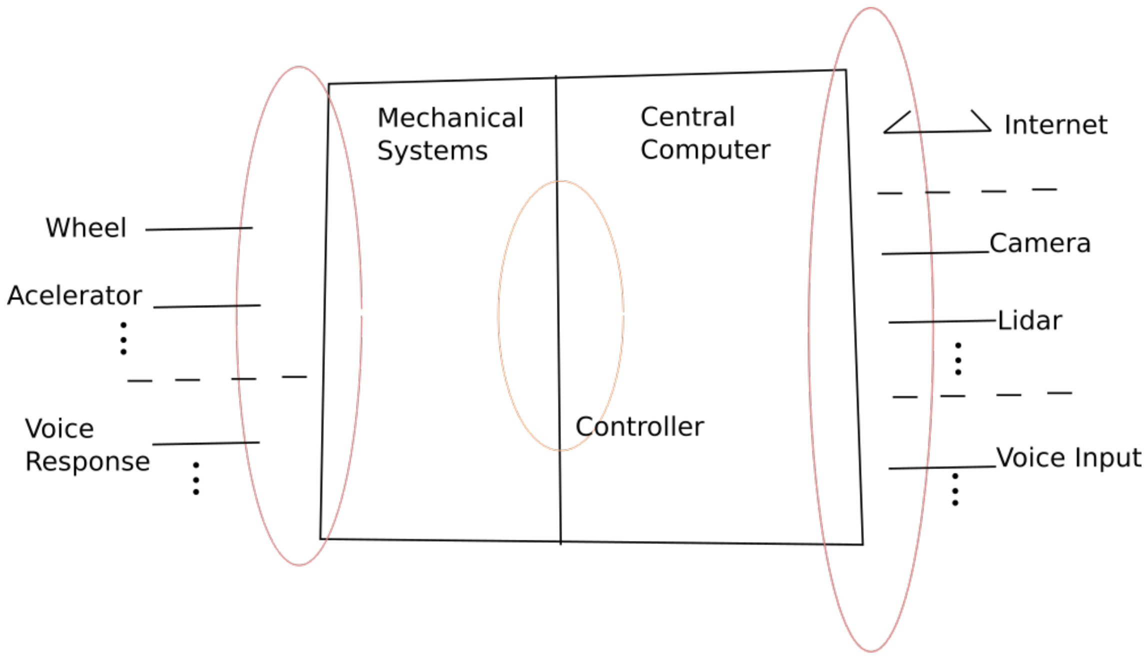

Figure 1: Self Driving Car

1 Abstract

We introduce a grammar for a controlled natural language (CNL) to give imperative commands for an envisioned voice assistant route-planner for a self-driving vehicle. The utility of the CNL is that it is inductively defined by a grammar : thereby, the sentences it admits, parsed as Abstract Syntax Trees (ASTs), can be manipulated as mathematical objects amenable to verification techniques. Using the TOUCHDOWN data set to empirically motivate common idioms and phrases our grammar should be capable of parsing, we give a denotational semantics from our ASTs to a Linear Temporal Logic (LTL) formulas, essentially expressing sequences of states which are amenable as specifications to downstream applications whose goal is the verification of various aspects of a vehicles behavior. This work contributes to a large existing literature, connecting the somewhat disparate research spaces including CNLs, verification for natural language-controlled robots, and semantic parsing.

2 Introduction

A central question in the philosophy of language concerns how language relates to the world. That is, how do our semantical notions relate to the physical world we perceive? How do we internalize and externalize our experience with linguistic structures? This question manifests concretely in the problem of designing a voice directed command system for an autonomous vehicle.

2.1 Problem Statement

We imagine the most simplified vision possible vision of an autonomous vehicle, a computer controlling the mechanical components of the vehicle to navigate based of sensor data from the environment, as seen in Figure 1. It is natural to partition the environment internally and externally : whereby the external setting may be captured by cameras, LiDar sensors, and a myriad of other sensors, and the internal environment may consist of a microphone to capture verbal commands given by a human. The goal of this project is then to connect the intentions of the human to the actuators controlling the mechanical vehicle relative to their shared perception of the environment through the modality of language.

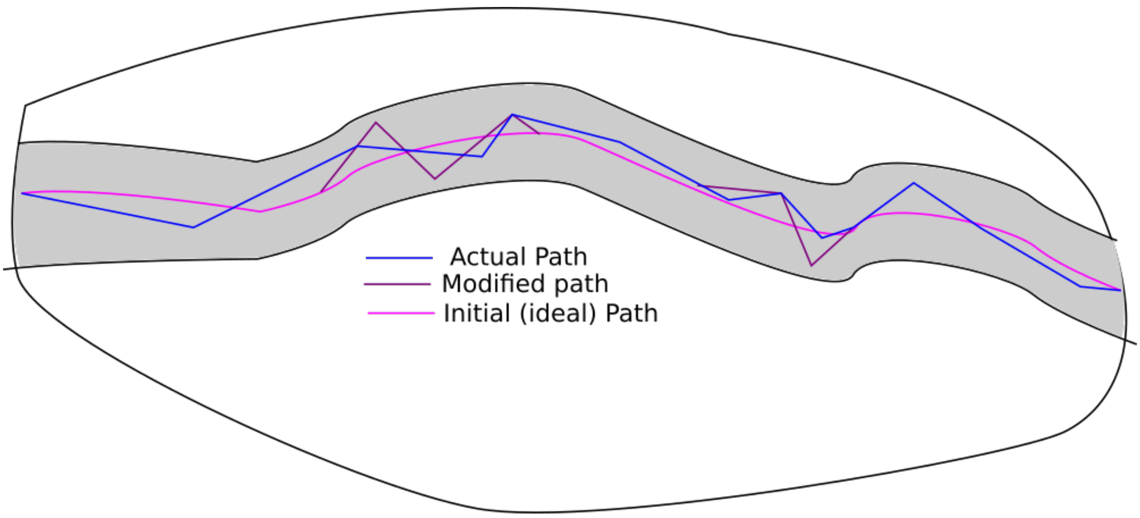

Figure 2: Vehicle Route

We can refine this picture, as is seen in Figure 2. Imagine making an initial voice command to the vehicle like “drive to San Francisco”, which the google voice assistant on my phone can already recognize, and plan the *initial path*. Then suppose that during the route, I get hungry and realize I need to refuel. I can utter “go to the grocery store after the next exit, and then go to fuel station, but stop at so I can take a picture of that crazy sign, first.” This is how a passenger might direct a human driver to take a *modified path*, and we take for granted that the passenger would find it the easiest to communicate this as if the computational agent is human. Finally, we know that neither the idealized nor the modified path can be met without local perturbations, and that the *actual path* should be optimized to conform to the modified path relative to some metric.

The many sub-problems under this grandiose vision are themselves grandiose. Training a neural network to accurately capture the meaning of natural language utterances (even in a domain specific setting), the ability to synthesize a path and a controller that meets the meaning specification, and the possibility of following the path in an unpredictable environment - these all have large communities working on them, and while there is certainly progress, a reliable end-to-end system should be treated with skepticism until some kind of empirical validation of such a system exists.

2.2 Overview

Perhaps the most pervasive question in the use and application of natural language technologies can be stated as follows : How does one optimize the system to provide for wide coverage of the domain while ensuring that system is robust? This question exemplifies the boundary of the verification-minded “formalist” and data-oriented “empiricist” camps in designing such technologies.

The statistical and machine learning methods applied to Natural Language Processing (NLP) tasks have produced impressive results over the past three decades. They take a more pragmatic approach : compromise robustness for wide coverage, as this means the tools will be usable and by non-experts. The belief espoused is that the machines should “learn” from (and possibly like) us. Somewhat orthogonal techniques prioritize the formal approaches of the computational linguistics communities. These methodologies are often more concerned with theoretical justification and explainability.

While practical tools are a goal, building practical applications often isn’t linguistically informative and therefore the empiricists’ goals shouldn’t override work on building the theoretical models which enable our understanding of the machines. Those in the formalist camp, prioritizing theoretically informed systems, seek predictable and well-defined behavior for specific problem domains. Yet, these systems fail to generalize without an explosion in complexity when presented with data outside their domain.

Natural language is difficult because it is both structured with respect to "rules", perhaps more descriptively titled *logical behaviors* which admit lots of predictability. Yet natural language continuously breaks or introduces exceptions to these rules, necessitating empirical and observational understanding. This makes it exceedingly hard to penetrate with exclusively the empirical or formalist approach. Many are led to wonder about the degree to which large amounts of linguistic data can be augmented with theoretical linguistic knowledge to create optimal and practical systems with respect to both breadth and depth of coverage of language phenomena. The ultimate question seeking compromise from both camps asks : how can we build machines which "understand" us (or at least our data), and which are comprehensible by us.

This problem acutely arises when trying to design a voice assistant in the domain of commanding controllable robots, specifically, autonomous vehicles. For the actions a vehicle takes, the motion and path decisions, must be formally specified and controlled via some computer system subject to mathematical formalism. Assuming the user directing the vehicle isn't aware of these formalisms, it is incredibly difficult to design a verifiable controller capable of dealing with the breadth of language one may encounter in the wild.

The instructions an arbitrary user gives are not subject to the same formalities the system requires. For her commands may leave out necessary detail ("go into the other lane" with multiple lanes on either side), say something wrong with respect to reality ("go into the other lane" on a single lane road), or give a command the driver should recognize as possible but bad ("drive directly into the car ahead"). Additionally, the driver may need to recognize many ways many users may say "the same thing", that is the same with respect to some semantic formalism. In a dual sense, the same utterance may admit two perfectly meaningful interpretations in two situations or contexts. The phrase "drive to the store with the dog" should account for whether the dog is inside of the car. It is obviously worrisome that a nefarious actor may somehow interfere with the controls at any stage by exploiting these manifold issues, and indeed many more. A failure to adequately deal with these circumstances in the vast majority of cases is not reassuring if one believes many verifiability criteria are critical for such technologies to see adoption.

We therefore analyze our "big-picture question" above in the following "sub-question" : how can one map the manifold ways of presenting information to an autonomous robot into a rigorous and formally verifiable kernel which the controller can understand? Our proposed solution is to build a semantic parser from natural language commands to linear temporal logic, whereby we can filter the many empirical natural language commands into a "canonical subset" defined by our CNL which are equivalent to (sets of) temporal logic formulas. We detail here both the progress to these ends, as well as the challenges.

3 Contributions

Initially motivated to give the system assurances against so-called substitution-based attacks, whereby impose "meaning equivalence" for synonymic expressions by imposing posterior conditions on the parse trees. Clustering via the tree structure to provide some the equivalence to meaningfully similar sentences was an initially enticing direction, as had been done with Komendantskaya and Heras' work on Machine Learning for Proof General (ML4PG) [19]. However, this work had the advantage that there are multiple large, well-maintained Coq libraries which were amenable to clustering. Successful clustering results could give rise to proof developers seeking suggestions in their developments.

Our use case, the design of a non-existent language, left us with the conundrum of an impoverished data set to train over. There was no empirical data source from which to observe "natural ASTs", and generating trees in an ad-hoc random basis would not likely provide real-world applicability. What has followed should be seen as a response to these constraints. Our intention was to find a data set suitable to give examples of non-trivial natural language utterances, in addition to finding a suitable semantic language with utility and applicability which is amenable to translation from sentences parsed by our grammar. Working from both the empirical and semantic directions seemed the most reasonable way to build a robust least prototype of a CNL .

The primary contributions of this undertaking so far are as follows :

- A Grammatical Framework (GF) grammar providing the definition of a CNL for suitable directives from a passenger to a driving agent
- A Haskell library mapping trees generated by our grammar a particularly well-behaved subset of LTL
- An Agda implementation of LTL with a standard semantic interpretation
- A refinement of the TOUCHDOWN dataset [10] suited to our needs of designing a better grammar

[TODO : ref sectionions]

We suggest that while each of these components are still relatively primitive, they define a pipeline which has potential to provide both theoretical insights to researchers and suggest possible practical steps that can be taken to constructing robust voice applications in industrial settings. In addition to discussing our own contributions, we give a relevant and comprehensive literature review that embeds our work in the context of ongoing studies about these topics.

This work should be seen as a stepping stone, specifically we view it as :

- A survey of existing literature about this problem
- A preliminary work attempting to fit in pieces of a solution to the problem
- A framework and prescription for how to fit this preliminary work carried out here with existing work from elsewhere in attempt to build a comprehensive solution

The feasibility of a realistic solution will doubtless rely on the future work of many collaborators and years of research, engineering, and testing. It is therefore naive to assume that many of the questions posed below are answerable.

4 Preliminaries

This research broaches many different fields, many of which were unknown to the author prior to this work. Indeed, voice assistants may encompass almost any natural language processing task, and autonomous vehicles are seen as one a premier emerging robotics technology (and certainly the most talked about in the popular zeitgeist).

Limiting the scope of work in this context can be challenging, as so many different tools and ideas can be seen as relevant. We therefore try to very explicitly narrow our focus to investigate how feasible it is to build a language for an autonomous vehicle that exhibits predictable behavior and also satisfies verification properties - this includes a determination to what extent the properties can even be stated. As the full development of such as a system is a grandiose vision, we hope to highlight many of the difficulties already arising, and also those one may anticipate.

The approach taken sets out to build a semantic parser, which, despite its primitivity, serves as a Petri dish through which many of the deeper questions in this space may be viewed.

4.1 Linguistic

4.1.1 GF, Parsers, and Personal Work

The questions of designing an idealized and expressive formal language, with roots in Frege [13], manifested more recently in the natural language semantic tradition of Montague [30], who proposed an interpretation of English in a typed higher order logic with a focus on quantifiers. Aarne Ranta, a student of Martin-Löf, attempted to reformulate Montague's work in an intuitionistic setting [34], thereby amenable to a natural treatment via computer programs [**ml79**]. In implementing a parser from natural language to a dependent type theory, Ranta discovered that the dual sugaring (pretty printing) transformation of a tree to a string could allow for a general mechanism of purely syntax-based translation. This work culminated in Grammatical Framework (GF) [35].

Grammatical Framework became a full research project, allowing for the simple specification of a parser using a statically typed programming language whereby the grammar rules could be seen as types. Separate concrete syntaxes cohering with a given abstract syntax allowed for language-specific parsing, sugaring, and translation. The GF "standard library", the Resource Grammar Library (RGL) [36], allows one to get off-the-shelf grammatical constructions for more than 30 languages, with English being the most comprehensively covered. The RGL therefore allows the grammar writer to focus on the semantic domain of the application the grammar is being developed for. In addition to this, one can embed a grammar as a Generalized Algebraic Datatype (GADT) in Haskell via the Portable Grammar Format (PGF) [1]. One can get runtime support for parsing and linearization directly in Haskell, in addition to manipulating the trees by pattern matching over them as Haskell programs.

A reflection on these historical developments reveals that GF is intimately tied to both the formal/informal distinction in addition to the syntactical and semantical approaches present in computational linguistics. These dual characteristics very much inform our problem as well. In the context of designing a voice assistant for, whereby one can give commands like "turn right after the woman with the big dog", we desire that the intensional belief a user has about her utterance is consistent with the extensional behavior of the vehicle. This can be done through an intermediary mapping to a formal semantic representation. Ensuring that the syntactic content of a speaker's (well-formed) utterance maps predictably to the logical form is important from the verificationist perspective : one wants to maximize the *syntactic completeness* of the system [25].

In a dual situation we briefly mention, one can imagine our voice assistant as giving the user feedback, responding with clarifications ("we will turn after the big cafe even though the other route may have less traffic"), questions ("do you mean this or that person?"), or even possible illocutionary directives ("we won't drive over the speed limit in a school zone"), requiring the computer to generate an utterance after it has made some internal determination. This internal deliberation must be a program, possibly expressed inside or outside our semantical space. It should be capable of identifying multiple routes in the clarification, multiple objects in a given state in the question regarding two people, or constraints based off external circumstances such as speed limits in school zones. In each case, the formation of a natural language utterance requires the computer to generate natural language which must conform to both a program's structure and behavior, but which also may be clear and recognizable to the user.

We recognize that there are many degrees of freedom in the both the syntactic and semantic formalisms chosen. With respect to parsing, one could choose a categorial grammar approach [11], or even forego using phrase-structure formalisms and use dependency grammars - of which there has been recent work in using dependency formalisms in conjunction with GF [37]. Additionally, many of our ideas should be applicable to robotics applications outside of the autonomous vehicle space, although syntactic, semantic, and data-specific nuances will have to be reconsidered for each domain.

Independently of *how* the robot determines a program whose meaning it needs to convey to a user, the property of providing a natural language utterance which fluently conveys meaning in a natural language to some native speaker is called *semantic adequacy* [25]. Determining a reasonable syntax and semantics for a controlled natural language should most certainly conform to the dual standards of syntactic completeness and semantic adequacy, if the voice assistant is to be held to any kind of regulatable standard.

4.1.2 Semantical Representations

We choose LTL as our semantic form in large part due to its relative expressivity for the kinds of verification conditions one might anticipate an autonomous vehicle needing to carry out, in addition to its ubiquitous appearance in the existing literature. Nonetheless, it is obvious their are many types of logical conditions LTL doesn't immediately support, and other logics, particularly ones which allow one to reason about space in its relation to time, would be an ideal direction to look. This line of research is probably more suited to people developing systems at later stages of development, where empirical observations may be collected in the wild. The nuances of where an autonomous navigator responding to a human agent can go wrong, and the most amenable set of verification conditions to prevent this, will ultimately have to be gained through trial and error.

Notions of Semantics We also note that the notion "semantics", having many connotations and interpretations in different fields, is subject to many interpretations. Here are some examples :

- In linguistics, semantics may be interpreted as intended meaning. Different theoretical notions of meaning may include a logical meaning, as in the case of Montague semantics, or a meaning as it arises in the use and context of culture, as is the case of cognitive semantics.
- In programming languages, the semantics of a syntactic entity most commonly means the mathematical behavior (denotational semantics) or behavior during execution (operational semantics).
- In statistical notions of semantics, one often seeks the ability of one to capture meaning via language use, most common in contemporary contexts, its practical uses. Frequently Word2Vec [28] is referenced in this context, although the advent of transformers in recent years has largely usurped this.

The problem presented in our work, of speaking to a machine, presents challenges in that it requires notions of semantics from disparate disciplines, which themselves have little overlap (at least as treated in the existing literature). This is because we are attempting to witness an utterance as a natural, native linguistic phenomena with an indented speaker meaning, a program whose syntax is defined via the CNL, and a statistical observation defined over some probability distribution of "sayable things". More concretely we ask :

- How is the speakers meaning interpreted as if intended to be understood by other native speakers?
- How does the speakers meaning manifest as a formal program a computer can evaluate?
- How can we identify a speakers meaning in a possibly infinite space of utterances and contexts in which those utterances arise, neither of which can be formally defined *a priori*?

Although the inter-relatedness of various semantic theories is a much bigger project than we can give space to here, it should be granted that problem we address forces one, both implicitly and explicitly, to try to grapple with them. We chose *the syntax of LTL* as the *semantics of our CNL* which is defined by filtering a "naturally observed" corpus to a primitive grammar. We propose to fit unseen utterances by fine-tuning a transformer-based language model to the corpus and grammar. We can then seek specific formalisms in which to analyze our problem domain :

- The meaning for a passenger-speaker can be analyzed in a variety of ways :
  - The meaning of an utterance is a logical formula following Montague's lead, substituting temporal operators for generalized quantifiers.

  - That the passenger's utterance should be determined as a speech act which carries illocutionary force and intention. The computer's response can be seen as conforming to or negotiating with the desires of the user, subject to the computers internal constraints and possible contextual information about which user may be unaware. Applying speech act theory in the context of human computer interaction has a long history [48].
- The meaning of the syntactic formula, can be interpreted in many possible ways
  - A type specification. In a case where temporal logic formulas are interpreted as types, Functional Reactive Programming (FRP), provides a functional programming context with which to interpret temporal formulas [45].
  - A (possibly verifiable) motion planner [39] [8] [21]
  - A dialogue state, in the envisioned Question Answer (QA) context, whereby the computer must provide feedback to the user based of contextual information
- The meaning from the mostly unseen utterances is given a canonical form, and the canonicalization process is a transformation via the vector-space and distributional notions of meaning implicit in an attention-based neural network

We don't intend to exhaust the list of possibilities here, neither in our description of the many meanings of "semantics", nor in how our taxonomy of semantics can be understood in the context of our solution to the problem of giving navigation commands to a autonomous driver. We intend to clarify some of the many subtleties and terminological confusions arising from many communities of researchers. We suggest that working towards a unified view of what kinds of semantic notions we want to deal in this particular domain may inform better solutions to the problem at hand.

Semantic Parsing The problem of semantic parsing consists of mapping natural language utterances not just to syntactic trees, but to semantic ones. A sub-field of Natural Language Understanding (NLU), building automated systems for mapping syntax to semantic forms can be traced back to Winograd's SHRDLU [47]. Although seen as a success at the time, SHRDLU was also incredibly brittle, and apparently led Winograd to step away from NLU, believing the problem too difficult.

The largest strain of contemporary interest in semantic parsers emerged during the resurgence applying deep neural networks to a variety of problems in NLP. An important observation, to view "semantic parsing as paraphrasing" [5], has greatly influenced the contemporary statistical approaches to semantic parsing. Much of this work has still used grammars as a central component in their pipeline, often to generate sentences randomly for the construction of a corpus to train with.

Towards the extreme of the data-centered perspective, it has been advocated to get rid of the parser in semantic parsers altogether. In [40], the authors naively takes for granted large public data-sets with syntactic and semantic forms, neither of which exist for autonomous vehicle syntax and temporal logic semantic formalism. Our approach takes for granted that the parser is one of the most controllable and easily understood components of a NL pipeline.

Semantic Parsing for Temporal Logics

> given an input English utterance, preprocess it to extract syntactical information, which may include part of speech tagging, dependency parsing, semantic role labelling, and so on. Then, enrich the input with these pieces of information. Finally, run an attribute grammar-based parser, or rely on some hand-made rules, to derive a translation into a target logical format. [9]

Brunello et al. give a thorough literature review of the many ways of translating natural language to LTL, indicating the interest and need of suitable semantic parsers in this domain.

We give a refined perspective on the problems below, deferring a formal treatment of LTL to the appendix [TODO : reference].

An interesting result published more recently attempts to translate between English and Signal Temporal Logic (STL) [17], which has the advantage of not just treating Boolean, but real-valued signals. From this perceptive, STL vs LTL can be understood as a quantitative versus qualitative way of analyzing events. The fact that STL gives the specifications a higher expressiveness in terms of how the order of events takes place, but also comes with a higher computational cost [ref needed], but more importantly, a potentially unnecessary complication for the system designer, The more granularity view of time may be unnecessary in many cases - our data set doesn't cover time at all (a defect, [reference below]), but even in a more naturally derived corpus, might only crop up as an "edge case" relative to other more important or likely phenomena that engineers may wish to capture.

4.2 Robot Motion Planning and Verification

The challenge of designing a system which generates robot control strategies from human language has to balance the expressiveness of task specification, complexity of environment, and provable correctness [4]. In this context, we assume that expressivity of the language itself should reflect the complexity of the environments, thereby being adequately descriptive. The criteria of correctness : that the language itself is well-represented in the LTL semantics - the system being is syntactically complete - is the focus of these investigations. Our work additionally, is the only work we know of which actually seeks autonomous vehicles as the central motivation, rather than more general robotics applications.

4.2.1 Temporal Logic for Robot Verification

We need a comprehensive view of the robot control problem as it pertains to temporal logics. Our considerations should include :

- The kinds of logical behavior one may wish to capture
- The sorts of missions we want our autonomous agent to accomplish
- The types of atomic grounded conditions one may want to include
- How do we model both the vehicle and the environment
- How do these logical behaviors interface with other components of the larger system

Temporal Logics Modal logics, specifically those dealing with stateful staging of events like LTL [2], Computation Tree Logic (CTL) [49], Signal Temporal Logic (STL) [3] , have been used extensively in the specification and verification of properties of robotics systems, including autonomous vehicles . As LTL is often seen as one of the "primitive" temporal logic, we chose it as a our semantic space despite its limitations (the lack of numerical precision, predicates for spatial relations, etc). We appreciate that future work will need to expand the scope of which logic (or possibly *logics*) the machine may use to verify behavior, in addition to the mathematical models most amenable to verification of a logical formula.

As regards the logical behavior, there are an array of logics available.

- Linear Temporal Logic (LTL)
  -
  -
- Metric Temporal Logic (MTL) go to the store within 5 minutes modal operators can express timing constraints
- Signal Temporal Logic (STL) the paths and models are now signals can be appended with a metric semantics, to show how well a formula satisfies
- Computation Tree Logic (CTL) follow the car in front of us

- CTL*
- Probabilistic Computation Tree Logic (PCTL) go to the store if its unlikely we'll hit bad traffic

Missions Types In the literature, there are two main properties of concern when specifying robot behaviors to be checked by models : *safety* and *liveness*. These are intimately related to the temporal modalities. Safety properties say "nothing bad ever happens", that is, a specification is satisfied globally by some model. Liveness conditions, on the other hand, mean that "something good eventually happens". An important theoretical result is that safety and liveness are expressively adequate : every property of interest can be decomposed into safety and liveness components [32].

This distinction is particularly relevant for our analysis, because we suggest that for the most part, directions given by a human (at least in our fragmented treatment) should be interpreted as liveness conditions. The expression of the desire to reach of a sequence of destinations, says that eventually we arrive at each such destination.

On the other hand, when we account for behaviors of the vehicle that are generally not intended to be instructed by the driver, these should be interpreted as safety properties. Obeying all traffic laws can be treated as a global condition that the neither passenger, nor an adversarial attacker should be able to override. Additionally, "comfort properties", like assuring the vehicle never accelerates too quickly or takes turns too quickly could also be encoded in this way. While there may be other mechanisms of enforcing or verifying that a vehicle meets these standards, we envision that the route planner (and the verifier) should treat the linguistic utterance as saying that do something good while simultaneously always never behaving badly.

We now discuss the missions that a user would want to instruct a vehicle to carry out, in the context of satisfying a stack of safety property preconditions. In [27], the authors empirically analyze an array of literature about robots and the types of missions they are typically employed for, filter out a subset of generalized LTL formulas which appear frequently, and design a tool PsA1M capable of building template missions over these formulas. We pay particular attention to what they call "core movement patterns", which include coverage (mainly what we're concerned with) and surveillance (which could be relevant, if, for instance, one wanted to design a autonomous-taxi that surveils a region of interest for passengers).

The coverage properties consist of visiting a set of locations, where various extra conditions like sequencing, ordering, and strictly ordering governs how. Specifically, given a finite (or incomplete) set of locations or events $\{l_i\}$ where $i \in \{1..n\}$ we can say a baseline coverage is of the form $\bigwedge_i F\ l_i$, which due to the commutativity of conjunction, doesn't distinguish between the order in which the locations are visited. To ensure that for every location l_i follows its predecessor l_{i-1}, we nest the locations in future temporal operators, essentially building a linked list $F\ (l_1 \wedge F\ (l_2 \wedge ...F\ l_n))$ with an extra F operator appended at every node. To induce the ordering, we can conjoin a predicate which restricts how the locations are sequenced by imposing the condition that location l_i must be visited prior to its successor l_{i+1}, namely $\bigwedge_i (\neg l_{i+1})\ U\ l_i$. If one wants to also ensure that the locations aren't redundantly visited, we can add the strictness condition. A bit confusing,

$\bigwedge_i (\neg l_i)\ U\ (l_i \wedge X\ (\neg l_i\ U\ l_{i+1}))$, this property ensures that one cannot revisit location l_i until after it has been initially visited and, in the next time increment, it hasn't been visited until its successor location l_{i+1} has. To see how these syntax trees are encoded in Agda, please see the appendix [TODO:REF].

As concerns directing autonomous vehicles, one anticipates that the user generally would want a sequential visit, with the order condition presumably but not necessarily intended in most situations. The strict order condition seems like it would need to be induced by the system almost automatically for efficiency reasons. We therefore chose to target the vanilla sequential visit with our Haskell program, although it merits a close investigation in which circumstances other logical predicates should be inferred, both based of specific language of the passenger and other contextual constraints.

Other movement patterns may reference past tense temporal operators, which may indeed prove very useful to verify that a users needs have been met, or the cases in which the route taken didn't conform to a command. We envision that developing a complete calculus of specification patterns in the specific domain of autonomous vehicles should very much inform how the CNL should be designed.

Environment and Grounding The historical development of logic in mathematics ultimately served to partition the mathematical expressions so that one could abstract the high level reasoning, the proposition and a proof structure, from the purely mathematical constructions. The interpretation of types in programming as logical propositions emanating from constructivist circles questioned this partitioning, allowing one to construct computer programs with mixed and indistinguishable "logical" and "mathematical". In the case of temporal logics for cyberphysical systems, however, the atomic propositions like "is-green-light", "truck-ahead", or "grocery-store" have no simple encoding in programming languages. This is because they aren't mathematical precise concepts, being empirically fastened to the environment in which the computer program operates. This is an incredibly important realization : how we witness the world is captured by some possibly faithful but incomplete mathematical abstraction.

For the atomic propositions in a temporal sentence to have meaning, we must ground them to the physical world. This grounding involves both the sensors collecting data from the physical environment, as well as the mathematical approximations we make of the environment. In [20], the authors review the many ways one can take propositions in various temporal logics and synthesize " correct-by-construction robot controllers", in the case there is no contradictory evidence about a specification's feasibility. Although this synthesis process exceeds the boundaries of the work, it is important to discuss because how one models the external environment and the events in it, should cohere to the internal environment - the natural language instructions. When building a system, reasoning about it from both directions is paramount.

Explicitly, how one chooses to approximate the external environment, and what a vehicle can and should *do* in it, should inform the constraints what one can say in the internal environment. In the case of a QA system where the car needs to inform the passenger of why the vehicle can't obey such instructions, or ask for clarification, the Natural Language Generation (NLG) phase will need to perform some kind of "reverse synthesis", and this process may well be much more difficult than the synthesis process to begin with.

The paper details three main ways of approximating the continuous external environment, modeled as a dynamical system specified by a first order differential equation, by a state transition system. These abstractions are from the dynamical system to a symbolic model represented by a transition system are partitions, motion primitives, and motion planners with trajectories. The partitioning generates a discrete transition system from a continuous state space, where actions represent movements between areas, and is contingent upon the dynamics of the differential equation. The motion primitives approach defines the primitives as maps between a different kind partition of the state space, and it is noted that they definitionally satisfy invariance and liveness properties, ideal for our application. To generate paths without regard to the dynamics of the system, motion planners deal with a geometric partition and builds robot trajectories based off where the robot is at a given time.

Once the abstraction from the external environment to a transition system has been realized, the translation of a logical formula into a Buchi Automata (or some other automata) can decided true or false relative to the Kripke model which has been generated. Unfortunately, the generation of an automata is doubly exponential in the size of the LTL sentence.

Using a tame fragment of LTL, Generalized Reactivity (1) (GR(1)), one may use gametheoretic approaches to that reduce the complexity of controller synthesis to polynomial in the size of the external environment's transition system [7]. The GR(1) fragment partitions the atomic propositions into two sets based of the environment and the state. Those formulas ϕ_e which represent environmental sensor data, and those propositions tied to the robot state, ϕ_s, grounded to actions and physical positions. The fragment deals with sentences of the form $\phi_e \implies \phi_s$, where both sets of formulas can be seen as a conjunctions of initial conditions, safety assumptions, and liveness conditions. In our example, this would mean that the command like "turn right at the fire hydrant" could enable (constructive) validation of the condition by ensuring that a camera takes a photo which shows a fire hydrant within some distance, then the car must be at a GPS certain coordinate on some map which has a fire hydrant, and that this should precede a state where the car sense a turn coupled with the steering wheel turning.

Unfortunately, the liveness conditions only allow for formulas of the form $\bigwedge_i G\ F\ \phi_i$ with ϕ_i restricted to using Boolean connectives. Nonetheless, by isolating environmental sensor data from the position certain and action primitives, and additionally separating out the liveness and safety properties, we see a clear methodology that can greatly simplify our specific problem. It is not known to the author if there has been work investigating synthesis coverage properties the fragment of LTL for the coverage properties, but it is well worth investigating, because despite the nesting of the F operators, we can certainly anticipate a reduction in the complexity of these types of formulas.

External and Other Constraints Linear logic, resource sharing, etc

The review [20] forgoes dealing with multi-agent systems,

5 Our Pipeline

As mentioned in the [TODO : Ref contributions], this work makes a proposal for how to create a general semantic parser from arbitrary voice commands to the coverage and sequencing subset of LTL, maximizing both coverage and robustness of the system. We outline the work done with respect to the following pieces, of our system, and conclude with the proposal which envisions a synthesis of all these pieces.

- A refinement of the TOUCHDOWN dataset [10]
- A refined version of the touchdown dataset
- A GF Grammar
- A PGF Haskell embedding of the Grammar
- An Agda implementation of LTL [TODO: see appendix]

5.1 Touchdown Data Set

The most comprehensive known data source relevant to this problem is the TOUCHDOWN data set [10]. The authors' used Amazon Mechanical Turk workers and an "interactive visual navigation environment based on Google Street View" based off images collected in New York City to generate English Language instructions to describe a route based off the visible environment to find a "touchdown object". Once the object is visible, the task is complete.

An intention of the work is a seemingly natural dataset which accounts for "resolving the spatial descriptions", but we focus only extracting the navigation instruction part of the task, knowing that the image classifier and the grounding mechanism is outside the scope of our work. Although the data comes equipped as JSON files with the text itself, a coordinate mapping instructions compatible with streetview, and other metadata, we just focus on the natural language text. The text samples consist sequence of constructions with visible cues, terminating with a description of where to find the "touchdown". A typical of the text is :

> Orient yourself so you are following the flow of traffic, Continue straight until you reach the intersection and take a left, You should see a red bus lane to your right

> and a bus stop, Continue straight until you are just past the bus stop, look to your right and you will see a tree with yellow foliage, click the base of the tree to find touchdown.

When filtering the dataset, we can choose to uniformly replace the last sentence reference the "touchdown" with the word finish. We first indicate some of the advantages of this data set with regards to our application.

- The size is reasonably large with approximately 10000 multi-command instruction sequences
- The descriptions are full of linguistic nuance and diversity
- The data is collected from multiple users, and is methodically produced
- The non-linguistic data may be of interest to those investigating how to grounding the utterances to the Street View panoramas

We imagine to a first approximation, these sequences are a great starting point when trying to build a robust system with respect to the linguistic environment.

- Finding the touchdown only coarsely approximates general navigating in a city.
- The workers aren't in a place they know, so everything the reference is in their immediate visual environment
- and this is a "short-term" task, it requires no long-distance navigation and reasoning
- Limited to NYC, hectic urban environments (also, daytime)
- Working with panoramas is not necessarily a great simulation to a real environment
- "They are not per- mitted to write instructions that refer to text in the images, including street names, store names, or numbers"
- No temporal reasoning (as spatio-temporal is assumed)

Ideally our data set would then have the properties

- Long and short-term tasks
- Different cities, different languages (this will be dependent on the context of the data collected), for instance, where there are dirt roads
- Updates over time (the user can update a context locally or globally) on the road
- Users with various degrees of contextual information Contextual information - street place, names (named entity recognition) accessible to google - people's names (mom's house)

That the data collection task in an objective way is inherently tied to the way we approximate it in collecting data, thereby limited by our experimental apparatus and assumptions in designing the data set.

What is the actual feasibility of this stuff?

how well does a given logic allow us to reason about a space of instructions. What is the logic grounded to, how it is verified , all of these may effect the choice of formulas

we can't just design a perfect language to capture our meaning - goes back to the wishful thinking of Frege, but we can try to approximate it

next intersection (and next left?) versus next gas station versus "next to" will be a store on your left with stars next to the name.

That having the data grounded is both incredibly beneficial, but also makes designing the syntax and semantics tricky.

6 GF Grammar

7 PGF Embedding

simplifying assumptions amgbiguity

7.1 Pipeline Proposal

The "sets of" clause references the inevitable ambiguity of parses even from a big enough parser, even if the size of the canoncial expressions is vastly smaller than the domain of expressions mapping to them.

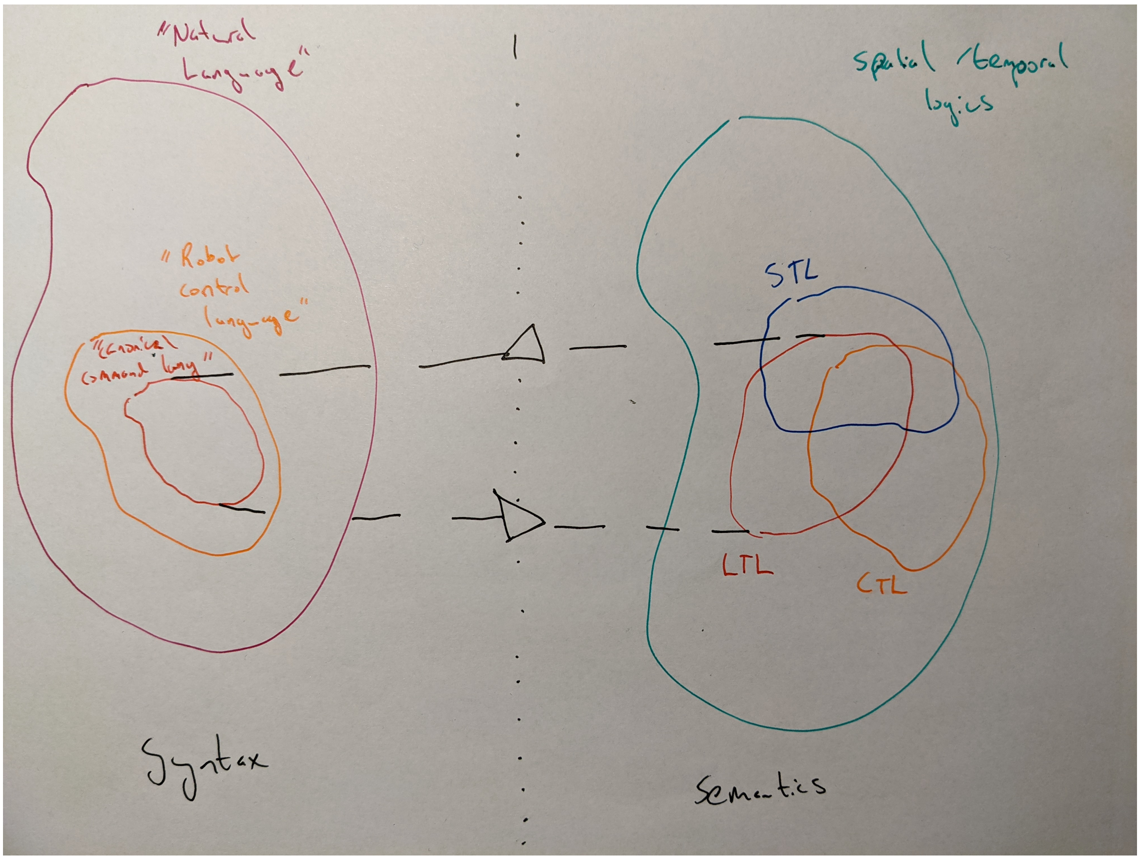

Figure 3: Language and Logical Spaces of Concern

We begin in Figure 3 with a high level overview of this semantic parsing system, whereby the space of natural language syntax can be mapped to some formal language semantic space (and possibly have some kind of inverse mapping). We note that "Natural Language", while an idealized notion, can be thought of the space of interpretable utterances. The relatively small subset of these utterances which one might give to a robot, labeled "Robot Control Language", is the ideal breadth our system would support, is still actually very large. We therefore applying another filter, to the "Canonical Command Language" which is inductively defined via some relatively thin set of grammar rules, which simultaneously generate and parse expressions in some logic. Although we target LTL because of its prominence in the literature and relatively straightforward implementation and interpretation, it should be noted that their are other temporal logics which may well be more expressive and better suited to the actual problem of synthesizing controllers.

Due to the recent influx of transformer based language models like Bert and GPT-3, we take for granted that the easiest way to target our "Robot Control Language" will be through fine-tuning one of these models, as shown in Figure 4. These transformers, trained on a separate corpus like Wikepdia, can be mapped to some suitable set of robot commands, even though these types of expressions will have a sparse presence in the corpus the model was initially trained on (presumably Marco will know more about this than me).

In this context, we can then further refine the language to something less natural, but more well-behaved. The whole proposed pipeline in Figure 5, indicates using the methodology as

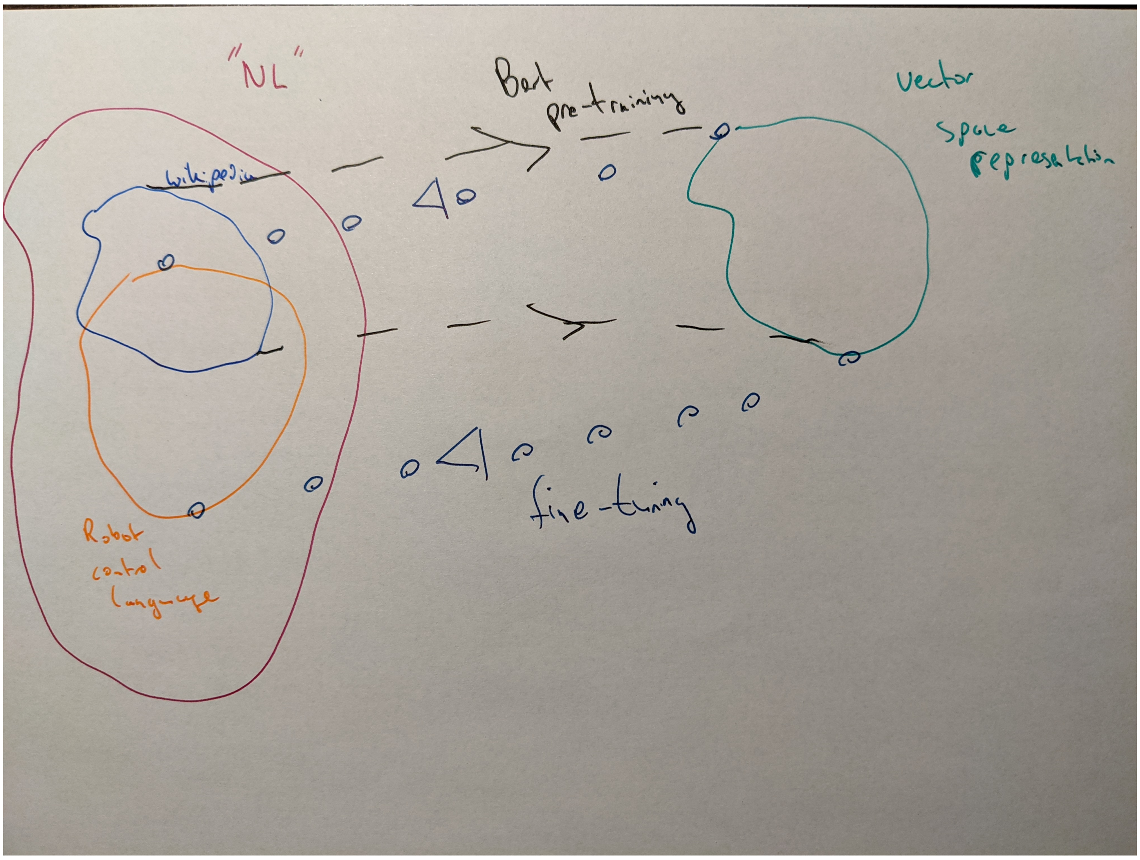

Figure 4: Transformer to Robot Control Language

used in [42], whereby the semantic parser should ideally be able to take any command from the Robot Control Language and turn it into a set of temporal logic formulas, distributed according to most likely interpretation.

Ideally, the downstream dialogue system should either be able to ask for clarification if two formulas are determined to be of some relative likelihood, reject a formula that is not determined to be achievable (for whatever reason), or synthesize a sequence of actions (and express those in the CNL) according to the possibly modified current path.

In theory, we can embed clauses which in turn reflect all of natural language : “Stop at the man who is watching the tv show on his phone about time traveler who goes back to the 12th century Mongolia, whereby the man, not speaking Mongolian ...” This is clearly outside the boundary of what the robot control language should support, and ideally would be accepted or rejected by the computer prior to the commands completion depending if there was a man looking at a phone. Our parser currently accepts strings in our primitive canonical language, designed in Grammatical Framework (GF), such as :

```
p "drive to the store , turn right and stop at the dog"

MultipleRoutes And (ConsPosCommand (SimpleCom (ModAction Drive (MkAdvPh To
(WhichObject The Store)))) (BasePosCommand (SimpleCom (ModAction Turn
(WherePhrase Right))) (SimpleCom (ModAction Stop (MkAdvPh At (WhichObject The
Dog))))))
```

However, we may envision our system being able to accept an expression in the Robot Control Language like “hit the petal till we reach the store, hang a right, and halt when you see a cute little puppy”. We could certainly adjust our parser to accomodate this, but it would be

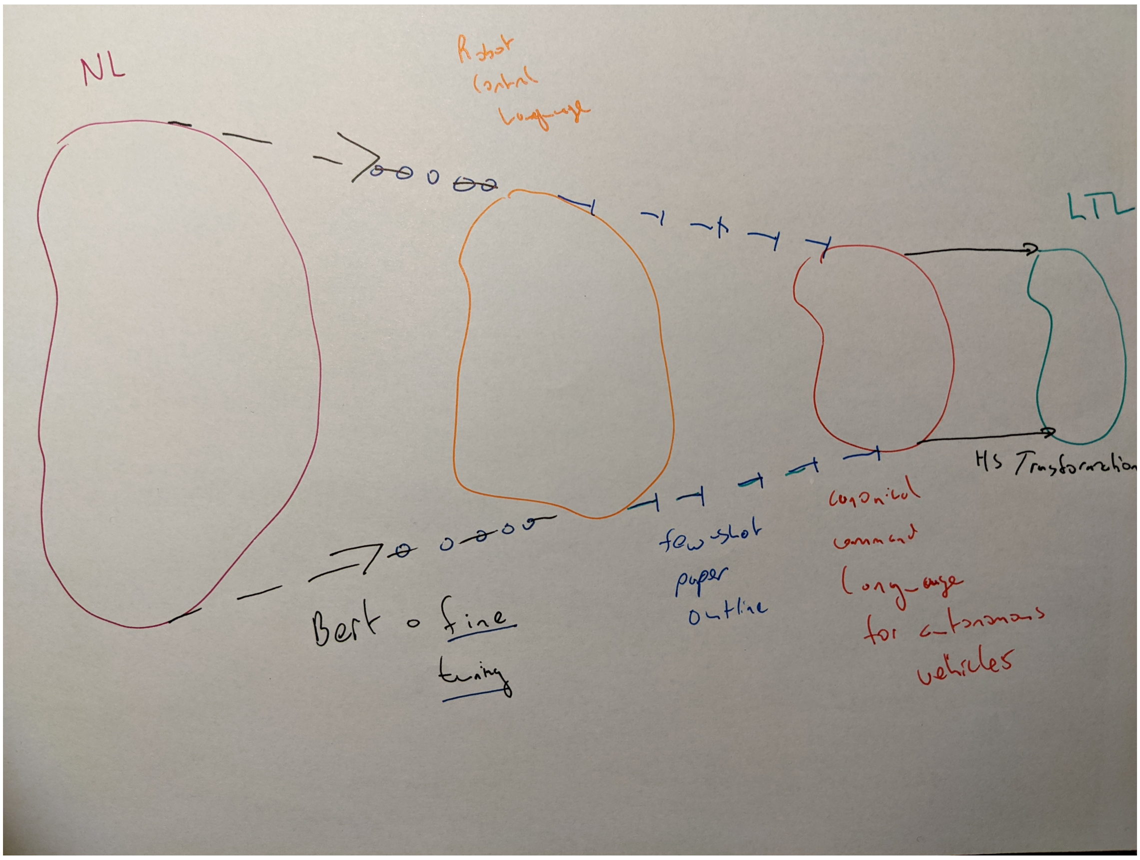

Figure 5: Pipeline from NL to LTL

one of many possible edge cases unlikely to be uttered. To accomodate many more such edge cases would cause an exponential blowup in the parser size (thereby slowing down parsing), but more importantly, cause the programmer a headache in building the parser, and then mapping the NL ASTs to a LTL form. If we treat F as the operating expressing the existence a future state, X as the next state, and G meaning the universal future, our desired LTL formula would most likely treat this as $F\ (store \wedge\ (X\ turn\_right\ \wedge\ (F\ (G\ dog))))$, although we propose that the actual grounding of these to images or controllable actions to some downstream system.

LTL has been a popular logic for specifying controllable robot behaviors, particularly with respect to verification of their behaviors. In [39], Rizaldi et al. prove logical correctness of a motion planner with respect to LTL formulas over maneuver automata formulas in the Isabelle/HOL theorem prover, a non-dependent cousin of Agda.

They adopt a modified semantics to [12], where they consider multiple paths instead of one. They use a model checker to generate the plan

We are choosing to deeply embed LTL in Agda for a few reasons, although the syntax of the embedding could easily be translated to any other dependently typed theorem prover, and with a little more effort probably any functional programming language. The composition of a "weakly verified" natural language front-end with a formally verified back-end such as in Rizaldi's work would pave the way for a fully verified, utterance-to-vehicle-path pipeline for the autonomous vehicles.

The big question to address is what kind of verification conditions the natural language component should be subjected to, and what kind of attacks would be most important to preemptively anticipate. Substitution based attacks [41], for instance, have been consistently emphasized throughout our discussions so far. The question is, *where* in the pipeline it would be best to filter out the vulnerabilities, as well as *how*.

One possibility would be to define words modulo equivalent meanings using Wordnet [29] in the syntactic phase, either via training [38] (presumably during the fine-tuning to the Robot Command Language or our "canonicalization" from that). It has been suggested that Bert is already relatively robust against such attacks [16], but we nonetheless feel that even higher sensitivities of robustness may be better done at other phases in the pipeline.

Alternatively, one could just map these equivalent Wordnet forms to equivalent parse trees using the Portable Grammar Format (PGF) Haskell library, which essentially deeply embeds a GF grammar into a Generalized Algebraic Datatype (GADT).

For instance, if we abstract over all abstract syntax trees for our grammar using this library, we can define the following Haskell functions to equate a "female human" with a "woman".

```
treeMapfemalePersonIsWoman :: forall a. Tree a -> Tree a
treeMapfemalePersonIsWoman (GModObj GFemale GPerson) = GWoman
treeMapfemalePersonIsWoman GWoman                    = (GModObj GFemale GPerson)
treeMapfemalePersonIsWoman gp = composOp treeMapfemalePersonIsWoman gp
```

There has been work integrating multiple language Wordnets with GF [44], so it would presumably be easy to integrate with our system, depending on how large we want the grammar to get.

As it is unclear what the best direction for this is, and how the attacker model in the context of an autonomous vehicle might work, all these decisions need to be made in the context of discussions within the group.

[Addendum before meeting :]

The TOUCHDOWN data set [10] seems like the most comprehensive and relevant data we'll find to fine-tune via one of these pre-trained models. Please see https://github.com/lil-lab/touchdown

The idea of domain specific pre-training can be traced to [15], where the authors introduce the concepts of *domain-adpative pretraining* and *task-adpative pretraining*, whereby this additional pre-training phase greatly improves efficacy of the LM on corpus and task data not well represented in the training data.

The language models have been show [22]

7.2 Syntax and Semantics

When designing a grammar, we can pretend that initially

san abstract syntax, we have the following considerations

That when we are conditioning our syntacitc model of emperical, noisy, and biased natural language data, so as to ideally generalize to unencountered phenomena.

A central insight ambiguity :

- Ontological semantic space. What are we trying to represent?
- Intended semantic space, the logical or formal system which our grammar will map to (via Haskell transformations)
- We want to account for some grammatical constructions via the abstract syntax, but outsource most of the grammaticality to the RGL
- Data source. How to conform to the data set in a way that's faithful but doesn't overfit (the overfitting can probably result in generating functions which are useless and either make our parser slow down or overgenerate)

While theres no clear way of relating the trade-offs, we can come up with some heuristics that shed light. Developing the "ontological design" allows one to capture the intuitive problem.

7.3 Ambiguity

What happens when we encounter ambiguity? For instance, in p ”go to the person with the dog .” The prepositional phrase ”with the dog” can either modify person (as an adjectival clause) or it can modify go (as an adverbial clause). Because the parser is designed to accommodate simple cases of both types of clauses, these ambiguities, even in simple sentences from our corpus, will grow quickly.

In the case of a vehicle, however, knowing the correct parse is dependent on the context in which the driver is going to the person : is the language grounded in the fact that there a dog in the car, or a person in the purview with a dog (or, most confusingly, perhaps both conditions are met, in which case more contextual information is required to disambiguate the correct parse).

For we can actually program the semantics to accommodate both scenarios, whereby

$F(manwithdog/\ GFinish)\ F(man/\ GFinish)/\ Gwithdog$

We can define our semantics to accommodate both interpretations, whereby the parses produce unique semantic conditions, and the LTL solver will have to see which condition is more easily satisfied. While this edge case may seem overly pedantic to consider, as one’s intuition might suggest the first case to be overwhelmingly more natural, the

8 Related Work

8.1 End-to-End Systems

8.1.1 Natural Language and LTL verification

Just as important as producing a well-formed and meaningful LTL formula, but not explored in our work, is the translation from a logical formula a trustworthy controller meant for navigating a complex environment. For instance, in [33] the authors indicate how to actually ground basic propositions from language to paths in a space, while our model, outputting formulas with non-grounded base predicates, is merely concerned with logical structure.

Similarly, in [8], the authors develop a Verifiable Distributed Correspondence Graph (V-DCG) model whereby LTL formulas are used to ensure grounded instruction sequences are consistent. This work builds on other work of Kress-Gazit et al. [23], whereby the The Situated Language Understanding Robot Platform (SLURP) allows translation of arbitrary natural language into LTL. They suggest an “ontology of common actions and the type of formulas that they produce” is of critical importance. Our work is directly focused on the *centrality of the grammar*, where a GF abstract syntax design allows one to give a precise ontology. We therefore see our work as a key intermediary phase when balancing formal and empirical interests. Kress-Gazit’s work is more concerned with the controllers generated as the end result of a pipeline where the intermediary grammatical structure may not be so relevant.

Our GF implementation, seeing the grammar as a necessary part of the verifiability (in that we can systematically map our sequences of commands to logical formulas representing sequences of states), also makes the possibility of supporting multi-lingual verifiability more immediate. Our system does not support this currently, but can easily be adjusted to so with the help of GF’s functors (roughly adapted from Standard ML’s functors) and the RGL. The lack of wide-coverage support of our grammar is possible to remediate through possible fine-tuning of a large language model to a data-set which coheres to the language our GF grammar generates, and we detail this in our discussion below [TODO : link].

Another approach seeks to train a natural language to LTL planner using both NNs and reinforcement learning [46]. Their work also uses a simultaneous CFG to generate *semantically inadequate* sentences with corresponding LTL formulas from which they can direct machines to follow the instructions, and then have users describe the robot behavior in a more natural form. Despite this, their approach uses the machine to generate sentences and corresponding situations, most of which are “nonsense” and need to be filtered out, thereby leaving the narrations upon which their system leaves devoid of a genuine empirical data source. In addition, their corpus only contains 266 words, still not the size one would need for our system. Finally, our suggested use of a pre-trained language model fine-tuned to the semantic parsing task gives us more flexibility in that the neural network and the verifiable grammar and semantics in the kernel could allow us to focus on the problems of breadth and depth somewhat independently.

The same group, in [21], explore the most general possible end-to-end utterance to planner pipeline without intermediary states, namely, a symbolic representation. While this “cutting out the middle woman” mentality may be an idealistic long-term vision, it makes the system much too much of a black box - even though they are able to reason about their system’s behavior through the use of attention maps. For the fine-tuned verification conditions about the linguistic utterances our work explores, the intermediate symbolic representations give a more explainable, predictable, and regulatable system.

Formal Requirements Elicitation Tool (Fret) fret

Use in aircraft, where there are many more controls, and the types of descriptions are inherently much more “structured”, i.e. the pilots are assumed to have expertise and aren’t just taking ad-hoc flights, but perhaps still don’t have engineering, verification, or logical knowledge to give precise LTL specifications

In [**fret**] they use the Prototype Verification System (PVS) theorem prover to verify that

[Problem] There are two major challenges in making structured natural language amenable to formal analysis: (1) associating requirements with formulas that can be processed by analysis tools and (2) ensuring that the formulas conform to the language semantics. [**fretish**]

—–

8.1.2 Tellex

Stephanie Tellex has written extensively about natural language inputs and interfaces with robots. Although she has not specifically written about autonomous vehicles, the domains have enough intersection to warrant careful consideration of much of her work, especially the recent stuff.

- Grounding with an intermediate symbolic state, no LTL, but possibly relevant for paper generally. She also cites [24], a seminal paper in this area

  > Instruction following is a supervised learning problem where the agent must predict a trajectory that would satisfy an input natural language command. [14]

- The review paper [26] making recommendations has a section on robustness, but this is mostly for the sake of allowing sharing of interfaces and efficacy, no mention of verification (which is what we’re primarily after)
- They design a NL -> LTL for drones that are grounded to actual landmarks [6]
- The group builds a trained pipeline that uses an object oriented template-instance methodology to generalize to different ontological categories in [18] [under review]
- In [31] build learn a semantic parser from NL to LTL (so that the language is grounded) where they collect executions of the LTL formulas in different environments using a weakly-supervised training method with reinforcement learning Part if the paper has to do with the execution of the command being dependent on the path taken by the robot executing the command, not just meeting the goal requirements, thereby giving a complexity bonus in comparison to previous work. She also evaluates the model on the [24] data set

8.1.3 Commericial

The public company Cerence [] is already designing voice assistants for autonomous vehicles, for which it has a large software stack between the voice processing to actual control of current automative components. In addition to its technologies, many of which aren't accessible to external researchers due to intellectual property restrictions, Cerence has contracts with large automakers [..]. It is therefore natural to inquire, what a small team with varied backgrounds and not nearly the same expertise nor experience within the technological team at Cerence can provide.

First, we believe that the focus on verification, insofar as we envision it, is unlikely to be of current concern at Cerence due to the fact that their products are still being developed, and the primary goal of producing a working product is likely to precedence over preventing non-existent hostile actors.

8.1.4 Alternative ideas

While the evaluation of machine learning systems provides assurances using different scores and metrics on different tasks assures one they may on average perform better than humans at certain tasks, the advent of adversarial attacks [43] with the intention of deceiving such a system by a hostile actor leads the system designer to desire, and possible require additional verification about the system's behavior. In the context of natural language processing (NLP), where data sources rely on strings of text, these attacks can focus an array of features from spellings of individual words to rearranging entire sentences []. So-called synonym attacks, which adversarially target the system at the lexical level, can cause traditional NLP models to [...] [].

Aside from the user experience being compromised by a system which has been adversarially afflicted, there is also a possibility of physical danger for the passenger and other people in the vicinity. As voice directed robots have many possible points of failure, we focus on two types of verification for our system. Rather than focus on breadth of language coverage, which ML language models excel at of due to their reliance on statistical modeling and tons of data, our system is narrowly focused as a proof-of-concept, from which it could either be extended by hand, or different components modified using other techniques and tools.

TODO : other ml stuff, like ravi's publication

9 Publications Description

9.1 Statistical (pre-trained) Language Models

- In [42] [under review], the authors show how, using a *synchronous context-free grammar* (SCFG) to define a minified CNL with a parallel and dually parsible semantic form, that one can use a large pre-trained language model as a front-end to filter a much wider syntax into the CNL. GF's expressivity is more expressive than the SCFG, at least based off a tertiary reading in the index, and therefore if we carved out a subset of commands to cohere with our LTL,
  our model would be amenable to a similar
- [16] [under review] claims that Bert is robust, analyzing claims of four papers, including the one which uses a wordnet attack

10 LTL Intro In Agda

References

[1] Krasimir Angelov, Björn Bringert, and Aarne Ranta. "PGF: A Portable Run-time Format for Type-theoretical Grammars". In: *Journal of Logic, Language and Information* 19.2 (Apr. 2010), pp. 201–228.

[2] M. Antoniotti and B. Mishra. "Discrete event models+temporal logic=supervisory controller: automatic synthesis of locomotion controllers". In: *Proceedings of 1995 IEEE International Conference on Robotics and Automation*. Vol. 2. 1995, 1441–1446 vol.2.

[3] Nikos Aréchiga. "Specifying Safety of Autonomous Vehicles in Signal Temporal Logic". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 58–63.

[4] Calin Belta et al. "Symbolic planning and control of robot motion [Grand Challenges of Robotics]". In: *IEEE Robotics Automation Magazine* 14.1 (2007), pp. 61–70.

[5] Jonathan Berant and Percy 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, June 2014, pp. 1415–1425.

[6] Matthew Berg et al. "Grounding Language to Landmarks in Arbitrary Outdoor Environments". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 208–215.

[7] Roderick Bloem et al. "Synthesis of Reactive(1) designs". In: *Journal of Computer and System Sciences* 78.3 (2012). In Commemoration of Amir Pnueli, pp. 911–938.

[8] Adrian Boteanu et al. "A model for verifiable grounding and execution of complex natural language instructions". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 2649–2654.

[9] Andrea Brunello, Angelo Montanari, and Mark Reynolds. "Synthesis of LTL Formulas from Natural Language Texts: State of the Art and Research Directions". In: *26th International Symposium on Temporal Representation and Reasoning (TIME 2019)*. Ed. by Johann Gamper, Sophie Pinchinat, and Guido Sciavicco. Vol. 147. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, 17:1–17:19.

[10] Howard Chen et al. "TOUCHDOWN: Natural Language Navigation and Spatial Reasoning in Visual Street Environments". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[11] Juraj Dzifcak et al. "What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution". In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 4163–4168.

[12] Georgios E Fainekos, Hadas Kress-Gazit, and George J Pappas. "Temporal logic motion planning for mobile robots". In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE. 2005, pp. 2020–2025.

[13] Gottlob Frege. *Begrisschrift*. Halle, 1879.

[14] Nakul Gopalan et al. "Simultaneously Learning Transferable Symbols and Language Groundings from Perceptual Data for Instruction Following". In: *Robotics: Science and Systems XVI, Virtual Event / Corvalis, Oregon, USA, July 12-16, 2020*. Ed. by Marc Toussaint, Antonio Bicchi, and Tucker Hermans. 2020.

[15] Suchin Gururangan et al. "Don't Stop Pretraining: Adapt Language Models to Domains and Tasks". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 8342–8360.

[16] Jens Hauser et al. *BERT is Robust! A Case Against Synonym-Based Adversarial Examples in Text Classification*. 2021. arXiv: 2109.07403 [cs.CL].

[17] Jie He et al. *From English to Signal Temporal Logic*. 2021. arXiv: 2109.10294 [cs.CL].

[18] Eric Hsiung et al. *Generalizing to New Domains by Mapping Natural Language to Lifted LTL*. 2021. arXiv: 2110.05603 [cs.CL].

[19] Ekaterina Komendantskaya, Jónathan Heras, and Gudmund Grov. “Machine Learning in Proof General: Interfacing Interfaces”. In: *Electronic Proceedings in Theoretical Computer Science* 118 (July 2013), pp. 15–41.

[20] Hadas Kress-Gazit, Morteza Lahijanian, and Vasumathi Raman. “Synthesis for Robots: Guarantees and Feedback for Robot Behavior”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 1.1 (2018), pp. 211–236. eprint: https://doi.org/10.1146/annurev-control-060117-104838.

[21] Yen-Ling Kuo, Boris Katz, and Andrei Barbu. “Deep compositional robotic planners that follow natural language commands”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 4906–4912.

[22] Jinhyuk Lee et al. “BioBERT: a pre-trained biomedical language representation model for biomedical text mining”. In: *Bioinformatics* 36.4 (Sept. 2019), pp. 1234–1240. eprint: https://academic.oup.com/bioinformatics/article-pdf/36/4/1234/32527770/btz682.pdf.

[23] Constantine Lignos et al. “Provably Correct Reactive Control from Natural Language”. In: *Auton. Robots* 38.1 (Jan. 2015), pp. 89–105.

[24] Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. “Walk the Talk: Connecting Language, Knowledge, and Action in Route Instructions”. In: *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*. AAAI’06. Boston, Massachusetts: AAAI Press, 2006, pp. 1475–1482.

[25] Warrick Macmillan. “On the Grammar of Proof”. MA thesis. University of Gothenburg, 2021, p. 90.

[26] Matthew Marge et al. “Spoken language interaction with robots: Recommendations for future research”. In: *Computer Speech and Language* 71 (2022), p. 101255.

[27] C. Menghi et al. “Specification Patterns for Robotic Missions”. In: *IEEE Transactions on Software Engineering* 47.10 (Oct. 2021), pp. 2208–2224.

[28] Tomas Mikolov et al. “Distributed Representations of Words and Phrases and Their Compositionality”. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS’13. Lake Tahoe, Nevada: Curran Associates Inc., 2013, pp. 3111–3119.

[29] George A. Miller. “WordNet: A Lexical Database for English”. In: *Commun. ACM* 38.11 (Nov. 1995), pp. 39–41.

[30] Richard Montague. “The Proper Treatment of Quantification in Ordinary English”. In: *Approaches to Natural Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics*. Ed. by K. J. J. Hintikka, J. M. E. Moravcsik, and P. Suppes. Dordrecht: Springer Netherlands, 1973, pp. 221–242.

[31] Roma Patel, Stefanie Tellex, and Ellie Pavlick. “Learning to Ground Language to Temporal Logical Form”. In: (2019).

[32] Nir Piterman and Amir Pnueli. “Temporal Logic and Fair Discrete Systems”. In: *Handbook of Model Checking*. Ed. by Edmund M. Clarke et al. Cham: Springer International Publishing, 2018, pp. 27–73.

[33] Erion Plaku and Sertac Karaman. “Motion planning with temporal-logic specifications: Progress and challenges”. In: *AI communications* 29.1 (2016), pp. 151–162.

[34] A. Ranta. *Type Theoretical Grammar*. Oxford University Press, 1994.

[35] Aarne Ranta. “Grammatical Framework”. In: *Journal of Functional Programming* 14.2 (2004), pp. 145–189.

[36] Aarne Ranta. “The GF Resource Grammar Library”. In: *Linguistics in Language Technology* 2 (2009).

[37] Aarne Ranta, Prasanth Kolachina, and Thomas Hallgren. “Cross-lingual syntax: Relating grammatical framework with Universal Dependencies”. In: *Proceedings of the 21st Nordic Conference on Computational Linguistics*. 2017, pp. 322–325.

[38] Shuhuai Ren et al. “Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 1085–1097.

[39] Albert Rizaldi et al. “A Formally Verified Motion Planner for Autonomous Vehicles”. In: *Automated Technology for Verification and Analysis*. Ed. by Shuvendu K. Lahiri and Chao Wang. Cham: Springer International Publishing, 2018, pp. 75–90.

[40] Subendhu Rongali et al. “Don’t Parse, Generate! A Sequence to Sequence Architecture for Task-Oriented Semantic Parsing”. In: *Proceedings of The Web Conference 2020*. WWW ’20. Taipei, Taiwan: Association for Computing Machinery, 2020, pp. 2962–2968.

[41] Suranjana Samanta and Sameep Mehta. “Towards Crafting Text Adversarial Samples”. In: *CoRR* abs/1707.02812 (2017). arXiv: 1707.02812.

[42] Richard Shin et al. “Constrained Language Models Yield Few-Shot Semantic Parsers”. In: *CoRR* abs/2104.08768 (2021). arXiv: 2104.08768.

[43] Christian Szegedy et al. “Intriguing properties of neural networks”. In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2014.

[44] Shafqat Mumtaz Virk et al. “Developing an interlingual translation lexicon using WordNets and Grammatical Framework”. In: *Proceedings of the Fifth Workshop on South and Southeast Asian Natural Language Processing*. 2014, pp. 55–64.

[45] Zhanyong Wan and Paul Hudak. “Functional Reactive Programming from First Principles”. In: *Proceedings of the ACM SIGPLAN 2000 Conference on Programming Language Design and Implementation*. PLDI ’00. Vancouver, British Columbia, Canada: Association for Computing Machinery, 2000, pp. 242–252.

[46] Christopher Wang et al. “Learning a natural-language to LTL executable semantic parser for grounded robotics”. In: *CoRR* abs/2008.03277 (2020). arXiv: 2008.03277.

[47] Terry Winograd. *Procedures as a representation for data in a computer program for understanding natural language*. Tech. rep. MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC, 1971.

[48] Terry Winograd and Fernando Flores. *Understanding Computers and Cognition: A New Foundation for Design*. Addison-Wesley, 1987.

[49] Chanyeol Yoo, Robert Fitch, and Salah Sukkarieh. “Probabilistic Temporal Logic for Motion Planning with Resource Threshold Constraints”. In: *Robotics: Science and Systems VIII, University of Sydney, Sydney, NSW, Australia, July 9-13, 2012*. Ed. by Nicholas Roy, Paul Newman, and Siddhartha S. Srinivasa. 2012.