

On the syntax and semantics of voice assistants in autonomous vehicles

Warrick Macmillan

6th May 2022

Motivation

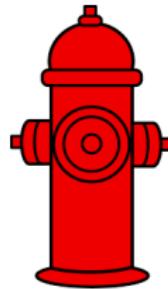
“Go to the grocery store after the next exit, and then go to fuel station, but stop by the fire hydrant so I can take a picture of that crazy sign, first.”

Motivation

“Go to the grocery store after the next exit, and then go to fuel station, but stop by the fire hydrant so I can take a picture of that crazy sign, first.”

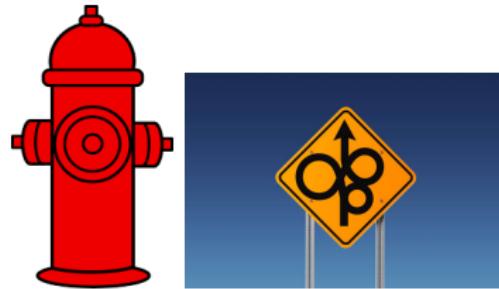
Motivation

"Go to the grocery store after the next exit, and then go to fuel station, but stop by the fire hydrant so I can take a picture of that crazy sign, first."



Motivation

“Go to the grocery store after the next exit, and then go to fuel station, but stop by the fire hydrant so I can take a picture of that crazy sign, first.”



Motivation

"Go to the grocery store after the next exit, and then go to fuel station, but stop by the fire hydrant so I can take a picture of that crazy sign, first."



Motivation

"Go to the grocery store after the next exit, and then go to fuel station, but stop by the fire hydrant so I can take a picture of that crazy sign, first."



Motivation

"Go to the grocery store after the next exit, and then go to fuel station, but stop by the fire hydrant so I can take a picture of that crazy sign, first."



Ambiguities

“Go into the other lane”

Ambiguities

“Go into the other lane”



Ambiguities

“Go into the other lane”



Ambiguities (cont.)

“Drive to the store with the dog”

Ambiguities (cont.)

“Drive to the store with the dog”



Ambiguities (cont.)

“Drive to the store with the dog”



This project is quite multifaceted, still in a somewhat primordial state (and therefore may be taken in many directions).

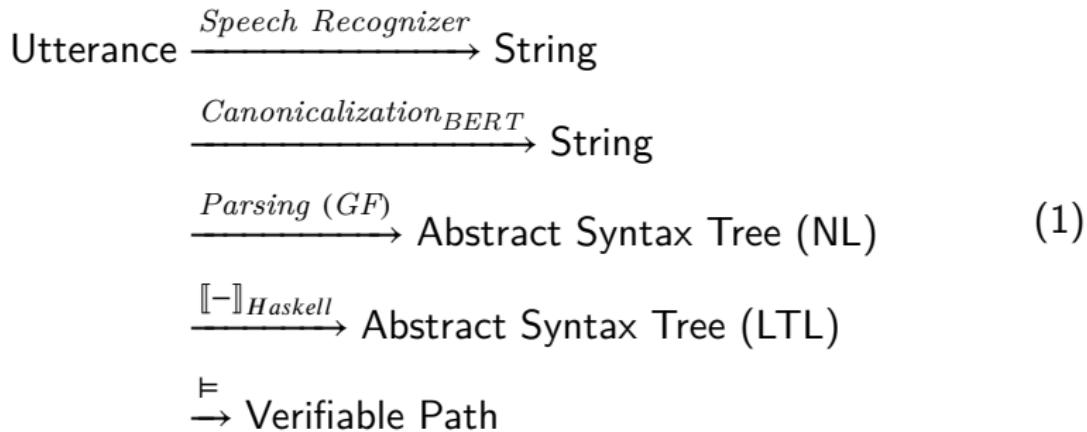
Goal : Design a controlled natural language which is

- Suitable as an “approximation” for a voice assistant for an autonomous vehicle
- Has a well defined semantics in temporal logic
- Seeking to balance breadth and depth of our system

Trade-offs

- Breadth : Wide coverage, usable by non-experts (where ML comes in)
- Depth : Well-behaved, verifiable (where FP comes in)

Ideal Pipeline



What we have so far...

String $\xrightarrow{\text{Parsing (GF)}}$ Abstract Syntax Tree (NL)

$\llbracket - \rrbracket_{\text{Haskell}} \xrightarrow{} \text{Abstract Syntax Tree (LTL)}$ (2)

$\xrightarrow{\models_{\text{Agda}}} \text{“Standard Semantics”}$

Example AST

“go to the store and turn left” \mapsto

CompoundCommand

- * And

- BasePosCommand

- * SimpleCom

- * ModAction

- * Go

- MkAdvPh

- * To

- WhichObject

- * The

- Store

SimpleCom

- * ModAction

- * Turn

- WherePhrase (Left)

Example Cont..

["go to the store and turn left . Finish ."_{AST}]_{Haskell}

F

```
(Meet
  (Atom "the_store")
  (X
    (Meet
      (Atom "left")
      (G (Atom "FINISHED")))))
```

Interpretation

Eventually be in a state where you are at the store. Directly after, you are in a state where you are left. From thereon, you are forever finished.

Functional Programming

Types, and type-checking is a fundamental way to separate the process of programming into two phases

- Static : Specification, Types
- Dynamic : Implementation, Programs
- Correct-by-construction mentality - “verification technique”
- Constraints are baked into user-defined types.
- Specification guides implementation. Functional programmers spend almost all their time thinking about types.
- Focus energy on high-level (abstract) details
- It is convenient to implement programming languages (i.e. write interpreters and compilers)

Grammatical Framework

- Functional programming language
- Implementing natural language parsers and linearizes
- Separation of abstract syntax and concrete syntax.
- Intended for translation, multilingual support
- One designs a grammar, Parallel Multiple CFGs (PMCFG)
- Chomsky Hierarchy : CFG < PMCFG < Context Sensitive Grammar
- Standard Library : Resource Grammar Library (RGL)
- Embedding in Haskell : Portable Grammar Format (PGF)

GF Example

BNFC (traditional treatment of fixity)

```
Lam. Exp ::= "\\" [PTele] "->" Exp ;  
App. Exp2 ::= Exp2 Exp3 ;
```

Abstract :

```
fun  
Lam : [Tele] -> Exp -> Exp ;  
App : Exp -> Exp -> Exp ;
```

Concrete :

```
lin
```

```
Lam pt e = mkPrec 0 ("\\" ++ pt ++ "->" ++ usePrec 0 e) ;  
App = infixl 2 "" ;
```

Haskell and Agda

Haskell

- One of the main FP languages - deep ties in Scotland and Sweden
- GADTs and pattern matching make for first-class tree transformations
- Defining and reasoning about logics and programming languages

Agda

- “Dependently typed Haskell”
- Interactive Theorem Prover
- Programs as proofs, propositions as types

Linear Temporal Logic

- Modal logic (temporal modality)
- Allows one to reason about sequential actions
- Verification for robotics systems
- Objective in reinforcement learning
- Other temporal logics (Signal TL, Computation Tree Logic, ...)
- Decidable

Complexity (and expressivity)

Propositional Logic < Temporal Logic <_{undecidable} First Order Logic

Temporal Operators

- $X\phi$: in the next state, phi holds
- $\diamond\phi$: exists a future state such that ϕ holds ($F\phi$)
- $\Box\phi$: ϕ holds for every future state ($G\phi$)



Where does ML come in?

Language Models

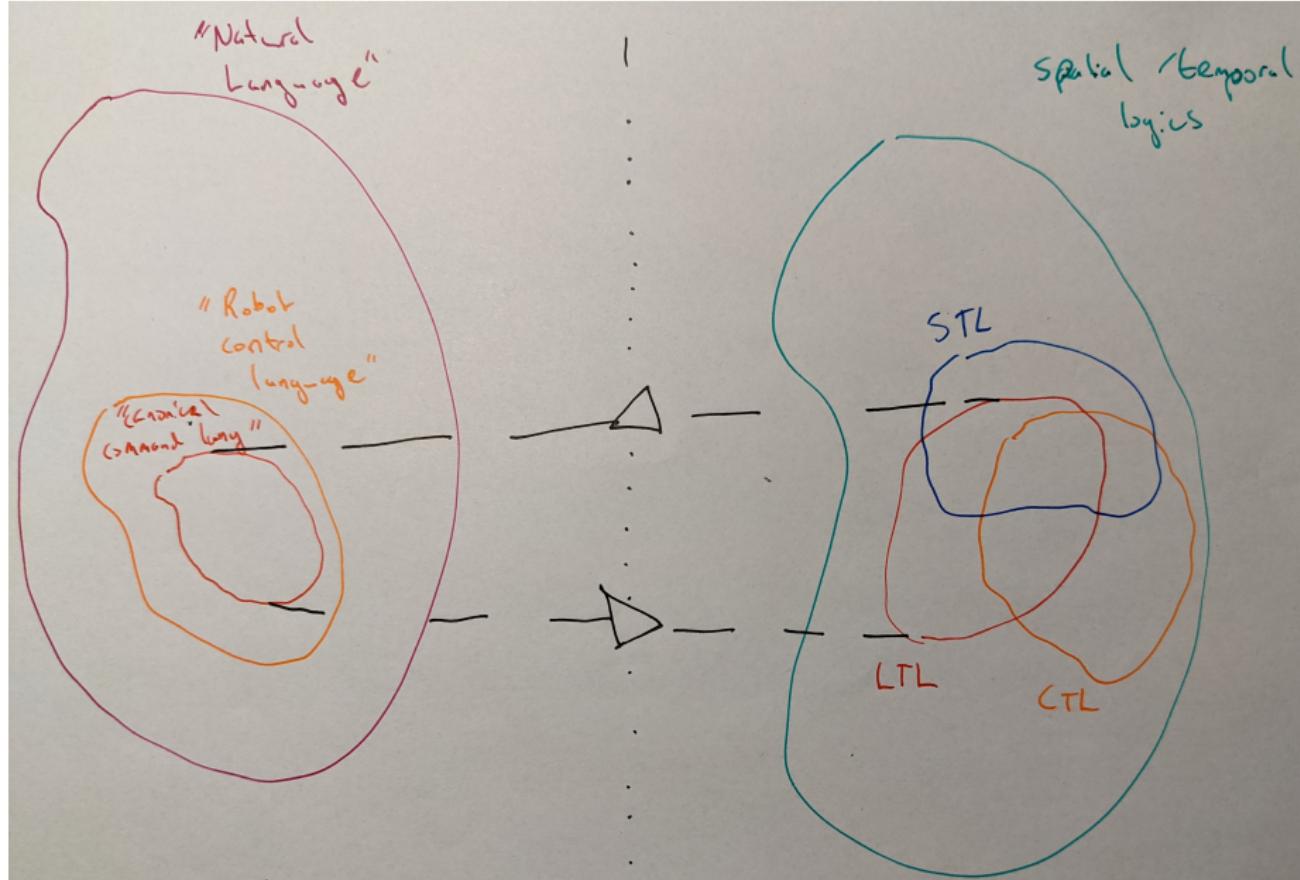
- Foundation models - future of NLP?
- BERT, GPT3, ...
- Pretrain and then fine-tune

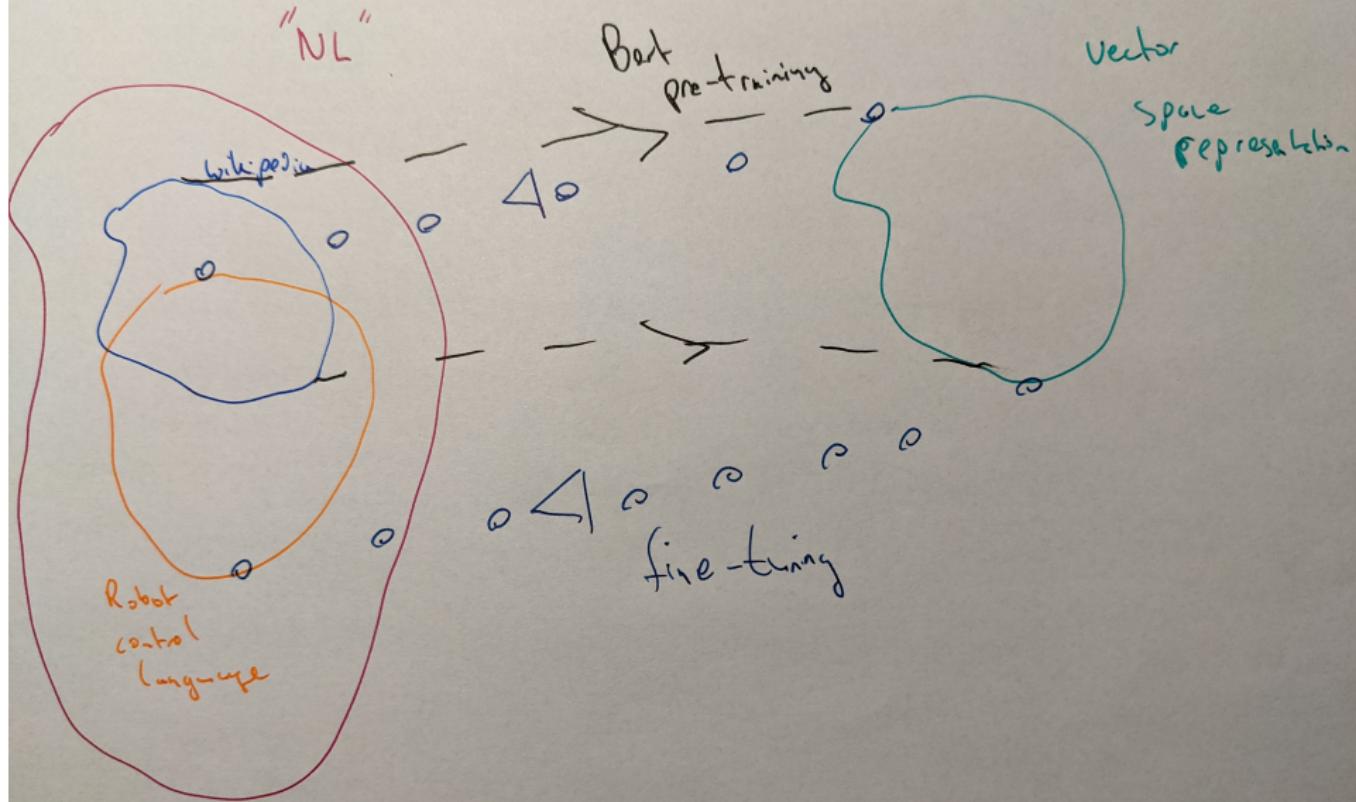
Ingredients

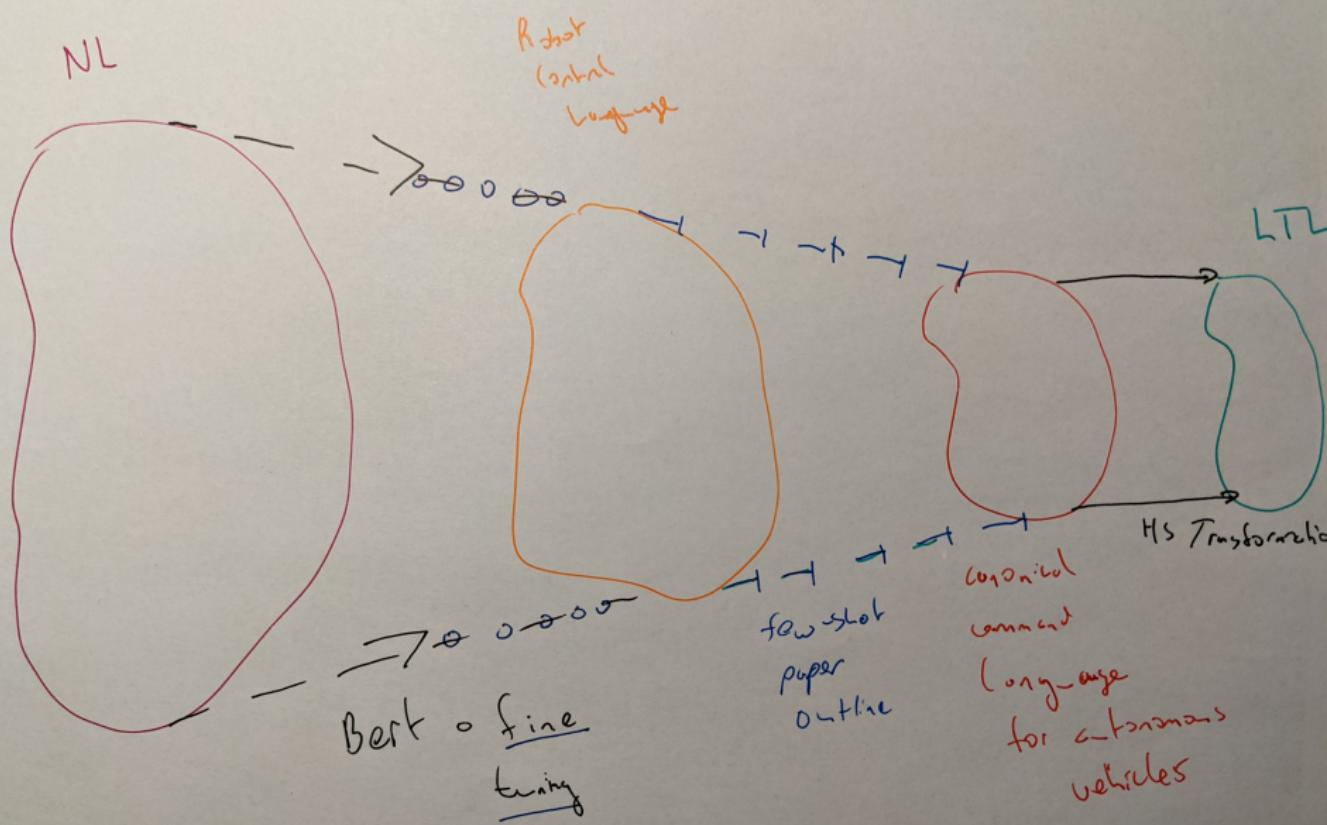
- Semantic Parser : NL Utterance -> Formal Language form
- Touchdown Data Set : 13000 sequences
- Few-shot semantic parsers paper (Microsoft Research) - open source Pytorch code

Idea

Integrate their technique and code with our parser and dataset







Future

- Expand parser itself
- Work on semantics , more expressive logic, etc
- Ground semantics to Touchdown location map
- Better data set?
- Working on a draft document, with references, of everything shown here.