# Translating English to Math

Dylan Orris

October 15, 2018

## Contents

# 1 Introduction

Over the past decade, natural language processors have become more and more commonplace. In late 2011, Apple introduced their digital assistant, Siri, to iOS. Siri was the successor to voice control, an iOS feature that allowed the user to interact with their phone using their voice, though in both an extremely limited fashion and rather unreliably. Siri was powered by a much better voice recognition model, and not only could process many commands into action, but could even ask for clarification based on context. Since this time, Google introduced the Google Assistant to its Android and ChromeOS platforms, Amazon released smarthome products powered by Alexa, and Siri has been ported to MacOS and WatchOS. All of these digital assistants rely on understanding the commands of users to be useful, and the only way this is possible is through natural language processing.

From Q2 2017 to Q2 2018, smart speakers, the quintessential aspect of the smarthome to many, had shipments grow 187%. This growth is expected to continue in the coming years, with smart speakers and the smarthome technology they may control becoming an enormous industry. As this industry expands, there will be greater and greater access to these systems among the general population. What if it was possible for a student to ask their smart screen why a mathematical claim was true, and that system could respond by displaying the proof? Smart devices offer more than simply convenience, they could become an amazing source of educational opportunities which are not otherwise available for many.

While we will not be using smart devices in this project, we will be creating a front end interface for students to more easily investigate mathematical questions, without the barrier of language.

## 1.1 Problem Statement

Automated theorem proving systems are a powerful tool which can allow a student to check their work or a researcher to prove a claim which could be extremely time consuming if done by hand. Unfortunately, these systems are not trivial to use, with the use of any such system requiring the user to learn the proper method for input, which will also differ between different such systems. As a result of this, it can be easy to be put off of these systems, and once one learns one such system, it is easy to be stuck using what is already known, even if better options may be available. A wonderful

solution to this problem would be leveraging computational power to convert a human language request, i.e. "Prove that the length of the third side of a triangle is determined by the two other sides and the angle between them" into a language the proving system will understand.

## 1.2 Project Aims

This project has two separate but interconnected goals; the production of a natural language processor, and the creation of a basic automated prover. The processor will:

- Serve as a front end for any automated prover.
- Be fully functional without any understanding of symbolic notation or programming
- Be modular enough to become the front end of any automated proving system.
- Accurately translate standard English to symbolic notation.

The prover will have far more modest aims:

- Understand basic questions in a limited scope, such as geometry or arithmetic.
- Take symbolic input, and return the truth value of said statement.
- Display the steps of the proof to the end user.

As the two pieces are separate, the processor will not be limited by the prover, and may instead be joined with a more robust system by specifying what said system is. This will be necessary to convert the request properly, so that the processor will know which symbolic language to translate to.

### 1.2.1 Natural Language Processor

The processor will take a user's input in English, and determine the goal of the user based on the content of that input. The user will not be required to know anything, other than that the system is intended for mathematical queries and will be able to do nothing else. Once the input has been analyzed, the processor will convert it to the proper symbolic language for the automated prover.

#### 1.2.2 Automated Theorem Prover

The prover will take a mathematical claim, in a symbolic notation, and apply axioms, theorems, and other knowledge as necessary to prove or disprove the claim. If the prover is able to use theorems in its proofs, it will be barred from using them in such a way as to give the appearance of circular logic. For example, when a user asks for a proof of the Pythagorean Theorem, the prover will be disallowed from using the Pythagorean Theorem as its justification for the truth of the theorem. Should the user ask for a proof of an axiom, it will be permissible to simply point to the axiom. The prover should not and will not be expected to explain an axiom to a user.

# 2 Previous Work

## 2.1 Natural Language Processors

### 2.1.1 Syntax and Semantics

### 2.1.2 Use in Information Retrieval

## 2.2 Automated Theorem Provers

### 2.2.1 Path Elimination

### 2.2.2 Domains of Knowledge

# 3 Software

## 3.1 Implementation Language

## 3.2 Natural Language Processor

### 3.2.1 Conversion to Transition Language

## 3.3 Automated Theorem Prover

# 4 Conclusions

## 4.1 Future Work