Ariel Camilo
Dr. Pablo Rivas

Formal Languages and Computability

Final Project

Abstract:

This project takes a look at some of the methods that we have used for deconstructing simple sentence structures, and how a DFA might be useful for determining which sentences that can be considered grammatically correct.

Introduction:

While studying the textbook chapter on grammars, I decided to read more on **learnrpolognow.org**. One of their pages describes context free grammars, which are a finite collection of rules which describes which sentences are grammatical. It describes a particular set of syntax which allows programmers (or anyone really) to break down common sentences in English and decide whether or not they are structurally correct. The basic structure is as follows:

s -> np vp

np -> det n

vp -> v np

vp -> v

det -> “a”

det -> “the”

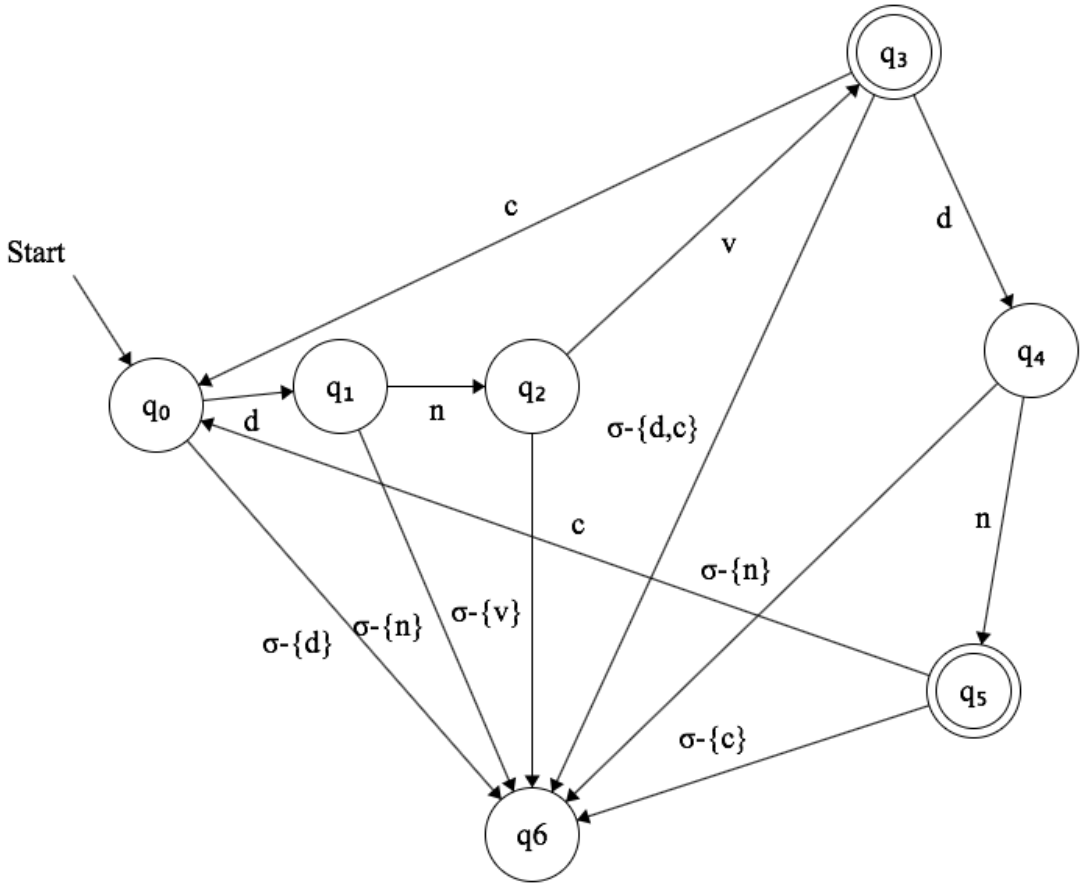
c-> “and”

c-> “but”

In this scenario the sentences (s) are constructed from noun phrases (np) and verb phrases (vp). Np's are constructed from determents and nouns. Vp's are constructed from verbs and noun phrases, or just verbs. The determents for my purposes here are simply the commonly used articles "a" or "the". I as well have added the conjunctions for allowing for multiple sentences to be conjoined.

Detailed System Description:

The system in place allows for users to input their sentence through the command prompt, and the DFA being implemented will simply return the inputted sentences, with an additional string added describing whether or not the sentence is grammatically correct. The DFA begins by breaking down the sentence into an array of words. The words are then checked against a list held by the program, in order to order them into an appropriate list of the previously mentioned grammatical terms. So for example the string “The woman shoots the man”, would be converted into “det n v def n”. This allows the DFA to then simply check the order of these grammar terms, in order to determine whether or not the given string is a grammatically accepted input. The diagram below details the DFA that the program implements.



We start off with the initial state q0, which immediately determines that if the first word is not an article, is sent to the failed q6 state. If an article is provided as the first word, then we can transition to q1 where a noun is expected. So on and so forth until we reach q3, which is an accepted state. An example of this would be, “The man dies”. However, most sentences usually implement another noun phrase, and so the q3 state can accept either a conjunction for another sentence, or another determinate for a more complete sentence. Any other input is treated as invalid, and as such we go to the q6 state. There are 2 final states q3 and q5. An example of a sentence which leads us to q5 as the final state would be: “The adult analyzes the officer”.

Requirements:

Any computer capable of running standard java programs, is just as capable of running this program.

Literature Survey:

There are countless programs out there which are machine-learning oriented, created to detect various forms of sentences, and can correct for syntactic errors that are more complex than the ones described above. Although not all, many of these are based off of previously accepted inputs. Advanced artificial intelligences are used to detect more and more patterns to more accurately describe which sentences are considered grammatically correct.

User Manual:

The system can be interacted through the command prompt. By having a text file populated with test sentences separated by a newline, the program can be executed with the following command *java CFGProgram < filename*. This will then kick off the program to analyze each sentence and determine whether or not the sentence is a grammatically correct one. Due to the obvious limitations of the vocabulary, there are many simple sentences that would not be properly detected, that would be if the proper vocabulary was added to the program. A quick glance at the *CFGProgram.java* file, will quickly tell one the lists of words and phrases currently accepted by the program.

Conclusion:

There are many limitations to the program. Some obvious and some less apparent. If one submits a proper sentence but the word simply isn't in the array of words the program keeps track of, then it will determine that the sentence is not grammatically correct. A larger list could be added to increase the number of accepted inputs. Another limitation would be that plural nouns and their respectively correct verb phrases would not be recognized. This limitation came about as the various exceptions to which plural nouns are considered grammatically acceptable, would easily be complex enough to merit it's own program, perhaps even implemented by its own DFA. This program represents an initial building block early programmers used to pave the road for more complex programs that are now used today, and that we all take for granted. Microsoft Word's autofill, and autocorrect being two of the more famous examples. The implementation of these types of programs allows individuals to focus less on the proper syntax and more on creating better content much quicker.

References /Bibliography:

There were multiple sources used to construct this program, yet non that could be considered scholarly.