LING571 – Hw3
Cocke-Kasami-Younger Parsing

1. Overview

I was **not** a member of a team when completing this assignment.

For this assignment I implemented the Cocke-Kasami-Younger parsing algorithm.

2. Challenges

The algorithm for calculating the table was not trivial, but it wasn't too difficult either. In contrast, the back-tracing aspect of the assignment was a real pain. Mostly because I am new to Python, and also because it took a long time for me to figure out how I could parse the table to get all possible interpretations, rather than only a single interpretation.

3. Learning Outcomes

Firstly, this week I realized the important of converting a grammar into Chomsky's Normal Form. Previously, I knew it was important, but I see now that it guarantees that a grammar can be represented as a binary tree. And this in turn, gives it certain properties that make it easier to implement Cocke-Kasami-Younger parsing algorithm, for instance.

Another interesting insight I had related to how to achieve back-tracing. In the end I used the heuristic that the pivot divides the sentence into two, and subsequent pivots further divided the sentence into two. Using this observation, I didn't need to record the originating cell, per se. (In other words, I didn't need an X-Y coordinate for back-tracing.) All I needed was the originating *pivot* index. From that, I could determine the (row, cell) coordinate of the preceding node in the tree.

To illustrate, if the start symbol S -> X Y is located at position (0, n) in the table and, at this position, we store the preceding pivot's index *i*; then we know that X -> Δ and Y -> Γ where Γ and Δ are other productions, then we can deduce that t X -> Δ can be found at position (0, *i*) and Y -> Γ can be found at (*i*, n). That is because the nodes at these position represent the valid subtrees that can be forced for the partitions of the sentence [0, *i*] and [*i*, n]; such that the span of terminals from (0, n) is covered. This can be extrapolated to complete the parse.

I think that, when presenting this course, it might be easier to understand the algorithm from this point of view. To think of the algorithm as being a table is not intuitive. Rather, the table is the caching mechanism used as part of the dynamic programming approach. It is more intuitive, conceptually, to think about the implementation as, primarily, a binary tree. The way I think about it now, the table is just a clever trick for storing all viable paths in **all** possible binary trees that could be used to interpret a sentence. This is pretty profound.

4. Closing Comments

I am very grateful for the extra day to complete this assignment. Having two assignment this week has been murder. Looking at my time-tracker, I can see that, during this week, I have spent 30 hours working on this course in addition to 33 hours of full-time work.

5. Completeness

I was able to complete the assignment.