Burak Tekin

Instructor’s Name

15 June 2018

Parsing - Project Report

Sentences:

There are 3 types of sentences are expected to be parsed. First sentence must belong to the language, the second one must only have an exact parsing tree and the last sentence must have ambiguity and must have more than one parsing trees. In order to these requirements, the following sentences are decided to be used in first grammar.

Zero parse tree: “the captain see the ship”

Exactly one parse tree: “the captain sees the pirate”

Ambiguous Sentence: “the captain sees the pirate with a telescope”

For the second grammar I decided to use a grammar consist of the letter symbols only as oppose to the natural language ones used in the first grammar. Second three sentences has been chosen for the second grammar like in follows:

Zero parse tree: “”

Exactly one parse tree: “”

Ambiguous Sentence: “”

Grammars:

For this project, it is decided to generate two different grammars which has different features. Both grammars have different rules and different non-terminals, terminals, productions. Furthermore, the type of the grammars also diverges from each other in the matter of whether being in Chomsky Normal Form or not. In the following part both grammar will be described in these sense.

Grammar 1:

In order to the requirements of the project, there has to be one grammar which is not in *Chomsky Normal Form (CNF)*. The first grammar is in CNF. Being in CNF is help us to use our grammar in CYK parsing without doing any preprocessing. Grammar in CNF always have 2n-1 steps to generate the string in the language, and having such a grammar also reduces the complexity to O(n^3) which is still greater than the forms not in CNF.

Grammar 1 has the following features:

- **Start symbol:** S
- **Terminals:** a, the, captain, sees, see, pirate, ship, with, telescope
- **Non-Terminals:** DET, N, NP, V, P, PP
- **Rules:**
  - **S --> NP VP**
  - **NP --> DET N**
  - **VP --> V | V NP**
  - **PP --> P NP**

Grammar 2:

This second grammar is not in CNF. So it has to be converted to CNF form. Below you can find this grammar's features and the steps to convert this grammar into the CNF form.

Grammar 1 has the following features:

- **Start symbol:** S
- **Terminals:** a, b, c, ε
- **Non-Terminals:** A, S, B, C
- **Rules:**
  - **S -> A S A | a B**
  - **A -> B | S**
  - **B -> b | ε**
  - **C -> A S | A B c | c**

To convert this grammar into CNF form:

1. If there is a starting symbol on the RHS, then we have to set a new rule and assign starting symbol 'S' to the new rule.
2. **Remove ε-production rules:** We have to get rid off ε-productions after dealing with starting symbol problem if exists. ε symbol should be found in the grammar and it has to be removed. I almost handle this very challenging problem but it has to be fixed.
3. **Remove unit-production rules**: If we have a rule which has only one possible non-terminal symbol(s) on the right hand side, then we have to get rid of it as well. To do this I add the rules of the corresponding productions of the symbol placed in RHS. In order to my attempts to eliminate this problem not showed any particular problem.
4. **Remove long-production rules:** If there are some rules exist with longer than 3 symbols has to be replaced by shorter rules. This also works in my implementation. I am looking for the rules longer than 2 units and try to modify them by regrouping symbols.
5. **RHS-one-terminal:** If a rule exist with a one terminal symbol on the RHS, we have to move all terminals to productions.

CYK-charts for each sentence:

Conclusions on whether the sentences belong to the language or not:

parse trees for each sentence (if they exist):

(NOTE: The results above with red colour could not be obtained from the last version of my code. Because of my busy schedule and my attempts to write a general version of a CYK parser, I needed more time to generate the results above. Hope it is acceptable enough for having a revision chance.)