CS 5000: Theory of Computability
Assignment 7
CFL Parsing with Cocke-Younger-Kasami and Earley Algorithms

Vladimir Kulyukin
Department of Computer Science
Utah State University

October 14, 2017

1 Learning Objectives

1. Chomsky Normal Form
2. Cocke-Younger-Kasami Algorithm
3. Earley Algorithm
4. Dynamic Programming
5. CFG Parsing

Introduction

This assignment has 2 problems. The first is on CYK. The scond one is on the Earley algorithm. In the first problem, you have to write some Python/JAVA code. In the second problem, you have to write a grammar for my JAVA implementation of the Earley algorithm. My apologies to the Py fans in this class. I don't have a Py implementation of this algorithm. If you feel up to it, you can implement the Early algorithm in Python for Problem 2.

Problem 1 (3 points)

Convert this grammar into CNF and implement the CYK algorithm to process strings with it. Your implementation of `CYK.py` or `CYK.java` should contain the boolean method `isInCFL(x)`, where `x` is a string. When `x` is in the language of the CNF grammar, the method returns `true`, else - `false`.

1. $S \rightarrow ABC$
2. $C \rightarrow BaB|c$
3. $B \rightarrow b|bb$
4. $A \rightarrow a$

Below are a few tests you can run to test your implementation. Implement these unit tests in your code. You may want to turn them into the methods `unit_test_1`, `unit_test_2`, `unit_test_3`, `unit_test_4`, and `unit_test_5`.

```
Input string: abc
Result = true

Input string: abbbabb
Result = true

Input string: abbc
Result = true

Input string: bbc
Result = false

Input string: aaabb
Result = false
```

Problem 2 (2 points)

Consider a drone flying over a grid of landmarks, e.g., A, B, C, etc. Suppose that our job is to design an natural language (NL) interface to the drone. Write 10 or so commands that a human operator, e.g., a search and rescue worker or a forest fire fighter, may want to ask of the drone. Here are some examples:

- `Which landmark are you over?`
- `Fly to A/B/C/D. What is your altitude?`
- `Could you fly higher?`
- `Fly higher/lower.`
- `Can you see A? Can you see B?`

Write a CFG grammar for the list of your commands and run the Earley Parser (EP) to parse your inputs and display the trees. My JAVA implementation of the EP is at `https://github.com/VKEDCO/EarlyParser`. This repo contains a few sample grammars. For example, `flight_grammar.txt` contains the grammar parts of which we analyzed in class. The file `ambiguous_grammar.txt` contains the formal grammar of arithmetic expressions we also analyzed in class.

The format of the grammar is pretty simple. The start symbol of the grammar is included inside the tags `<S>` and `</S>`. The variable symbols, one symbol per line, are placed inside `<V>` and `</V>`. The terminal symbols, one symbol per line, are placed inside `<T>` and `</T>`. Finally, the productions, in BNF form, one production per line, are placed inside `<P>` and `</P>`.

The class `Parser.java` contains several examples of how you can read the grammar from a text file and then parse a string with it. For example,

```
static void parse_example_01() {
   System.err.println("======== parse_example_01 ===========");
   String gfPath = "sample_grammar.txt";
   Parser epr = Parser.factory(gfPath, 3);
   ArrayList<ParseTree> ptrees = epr.parse("book that flight");
   epr.displayChart();
   epr.displayParseTrees(ptrees);
}
```

The first argument to the static method `Parse.factory` is a file path while the second argument is the number of wordforms in the input. Since we are parsing "book that flight", the number of wordforms is 3.

Save your CFG in `drone_cfg.txt` and save the parse trees dislayed by the EP in `my_parse_trees.txt`.

What to Submit?

Submit your implementation of `CYK.java/py`, `drone_cfg.txt`, and `my_parse_trees.txt` via Canvas.