2nd Partial Project

Computational Mathematics

October 10, 2016

1 Introduction

The Cocke–Younger–Kasami algorithm (also called CYK) is a parsing algorithm for context-free grammars (CFG), named after its inventors, John Cocke, Daniel Younger and Tadao Kasami [2]. It is considered a bottom-up parser that uses dynamic programming to build an AST of a given input string. The standard version of CYK works only on CFG given in Chomsky normal form (CNF).

In the worst case, this algorithm has a running time of $\Theta(n^3 \cdot |G|)$, where n is the length of the parsed string and $|G|$ is the size of the CNF grammar G. Therefore, it is one of the most efficient parsing algorithms in terms of worst-case asymptotic complexity, although other algorithms exist with better average running time [1].

2 Algorithm Description

The input is a context-free grammar in CNF G and an input string x. This means G will contain production rules of the forms $A \to \alpha$ and $A \to BC$.

The basic idea of this algorithm is to find for each substring w of x the set of all non-terminals (V_t) that generate w. This is done by building a data structure that will contain all the V_t that leads the creation of w. If at the top of this structure, the start symbol S is present, this means such a string x can be built using G. Therefore, w can be generated by G.

Listing 1: Algorithm 1

```
CYK (G < V_n, V_t, P, S >, x)
for i=0 to n−1 do
        T_{i,i+1} = ∅
    for A → a do
        if a = x_{i,i+1} then
                T_{i,i+1} = T_{i,i+1} ∪ {A}
                end if
        end for
end for
for m=0 to n do
```

```
        for i=0 to n−m do
        T_{i,i+m} = ∅
        for j=i+1 to i+m−1 do
                for A → BC do
                if B ∈ T_{i,j} and C ∈ T_{j,i+m} then
                        T_{i,i+m} = T_{i,i+m} ∪ {A}
                                end if
                        end for
                end for
        end for
end for
```

3 Deliverables

As part of this project, you and your team have to submit the following:

1. The executable file of your implementation using the programming language you like.
2. The source code of such an implementation.
3. The following testing case:
   - Given the grammar:

$$\begin{array}{ccccc} S & \rightarrow & A\ B & | & B\ C \\ A & \rightarrow & B\ A & | & a \\ B & \rightarrow & C\ C & | & b \\ C & \rightarrow & A\ B & | & a \end{array}$$

     Verify if $w_1 = baaba$ and $w_2 = ababa$ are both accepted. *The solution has to contain the generated AST.*

The solution has to be presented by the complete team at a date and time previously agreed on. Without this, your project cannot be marked. You should present before **October 26th**.

References

[1] John E. Hopcroft, Rajeev Motwani, Rotwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computability*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2000.

[2] Daniel H. Younger. Recognition and parsing of context-free languages in time n3. *Information and Control*, 10(2):189 – 208, 1967.