AUTOMATA FORMAL LANGUAGES AND LOGIC

Lecture notes
CYK (Membership algorithm)

Prepared by:
Kavitha K N
Assistant Professor

Department of Computer Science & Engineering
PES UNIVERSITY
(Established under Karnataka Act No.16 of 2013)
100-ft Ring Road, BSK III Stage, Bangalore – 560 085

Table of Contents:

Examples Solved:

CYK Algorithm

1. Introduction

The CocKe - Younger - Kasami algorithm alternatively called as CYK or CKY is a parsing algorithm for context free grammar named after its inventors, John , Daniel Younger and Tadao Kasami. It is a membership algorithm for Context free grammar.. It employs bottom up parsing and dynamic programming.is
It is used to decide whether a given string belongs to the language of the grammar or not.

CYK algorithm operates only CFG given CNF.
The worst case running time of CYK algorithm is Θ (n3. |G|)
Where n is the length of the parsed string and |G| is the size of the CNF

To check whether the string belongs to the grammar , we should construct a table (similar to table filling algorithm).
Construct a triangular table such that
-> Each row corresponds to the length of the substring
- Bottom row represents the substring of length 1
- second row from bottom row represents substring of length 2
- third row from bottom row represents substring of length 3 and so on
- top row represents the entire string ‘w’ length

For example , consider the string of length 5, $\mathbf{w_1\ \ w_2\, w_3\, w_4\, w_5}$
For this the table looks like

<table>
<tr><td>$\mathbf{X_{15}}$</td><td></td><td></td><td></td><td></td></tr>
<tr><td>$\mathbf{X_{14}}$</td><td>$\mathbf{X_{25}}$</td><td></td><td></td><td></td></tr>
<tr><td>$\mathbf{X_{13}}$</td><td>$\mathbf{X_{24}}$</td><td>$\mathbf{X_{35}}$</td><td></td><td></td></tr>
<tr><td>$\mathbf{X_{12}}$</td><td>$\mathbf{X_{23}}$</td><td>$\mathbf{X_{34}}$</td><td>$\mathbf{X_{45}}$</td><td></td></tr>
<tr><td>$\mathbf{X_{11}}$</td><td>$\mathbf{X_{22}}$</td><td>$\mathbf{X_{33}}$</td><td>$\mathbf{X_{44}}$</td><td>$\mathbf{X_{55}}$</td></tr>
</table>

Each cell will enumerate some variable, for example $\mathbf{X}_{11}$ should see the terminal with length 1, $\mathbf{X}_{12}$ enumerated with $\mathbf{w}_2$ Length.
To enumerate each cell you should compute utmost n previously generated sets,

For example, to compute the $\mathbf{X}_{12}$ you should make use of $\mathbf{X}_{11}$, $\mathbf{X}_{22}$. If you want to fill the cell , you should see the previous pair in the row below.
To compute $\mathbf{X}_{23}$ you should see $\mathbf{X}_{22}$, $\mathbf{X}_{33}$ to compute $\mathbf{X}_{34}$ you should see $\mathbf{X}_{33}$, $\mathbf{X}_{44}$ and so on.

Suppose we manage to get all the possible values of $\mathbf{X}_{ij}$, then it is quite clear that the string X belongs to L(G) iff
$\mathbf{X}_{in,}$ Contains the start symbol S, where n is the length if the string, (ie, the top cell should contain S in it)

2. Example:

1) Parse the string abba using CYK algorithm ,

Grammar:
S -> aSb | bSa | SS | $\boldsymbol{\lambda}$

Note:

If we want to fill $\mathbf{X}_{ij}$ we should see what is previously computed utmost n pairs so $\mathbf{X}_{ij}$ can be expanded as:

$\mathbf{X}_{ij} = (\mathbf{X}_{i,j}\,\mathbf{X}_{i+1,\,j}) \cup (\mathbf{X}_{i,i+1}\,,\mathbf{X}_{i+2,j}\ \ldots\ \mathbf{X}_{i,\,j-1}\,,\mathbf{X}_{jj})$

Solution:

Step 1:
Convert given CFG to CNF
Eliminate $\boldsymbol{\lambda}$ **production**
S -> aSb | bSa | ab | ba | S | SS | $\boldsymbol{\lambda}$
Eliminate unit production
S -> aSb | bSa | ab | ba | SS | $\boldsymbol{\lambda}$
There are no useless production

Conversion to CNF

S -> AB | BA | AC | BD | SS | $\boldsymbol{\lambda}$

A -> a
B -> b
C -> SB
D -> SA

Step 2:
CYK algorithm

4	S			
3	C	∅		
2	S	∅	S	
1	A	B	B	A
	a	b	b	a

1) **Strings of the length 1 can be generated by**
   A -> a
   B -> b
2) **Strings of the length 2 can be generated by**
   For AB
   S -> AB
   For BA
   S -> BA
   For BB it is ∅

3) **Strings of the length 3 can be generated by**
   a) **A . ∅** ∪ S . B = **∅ . SB (SB is generated by C)**
      C -> SB
   b) B .S ∪ **∅ . A**
      BS is not generated by any rule

4) **Strings of the length 4 can be generated by**
   A . ∅ ∪ SS ∪ CA
   S -> SS

The given string belongs to the grammar

Example 2:

S -> AB

A -> BB | a

B -> AB | b

String : aabba

5	**Ø**				
4	A	**Ø**			
3	S,B	A	**Ø**		
2	**Ø**	S,B	A	**Ø**	
1	A	A	B	B	A
	a	a	b	b	a

Length 3:

1) A (S, B) ∪ ∅.**B** **(S -> AB, B -> AB)**
   AS, AB , ∅
2) **AA** ∪ (S ,B) (B) (A -> BB)
   AA ∪ SB,BB
3) A . ∅
   ∅

Length 4:

1) **AA** ∪ ∅**. A** ∪ (S,B) (B)
   AA ∪ ∅ ∪ SB ∪ BB
   (A -> BB)
2) A. ∅ ∪ (S,B) ∅ ∪ AA
   ∅ ∪ ∅ ∪ AA
   =∅

Length 5:

A ∅ ∪ ∅ . ∅ ∪ (S,B) ∅ ∪ AA

= ∅

The string does not belong to grammar

Example 3:

S → AA | BC

A → BA | a

B→ CC | b

C → AB | a

W= baaa

4	S,A,C			
3	**Ø**	S,A,C		
2	A,S	B	B	
1	B	A,C	A,C	A,C
	b	a	a	a

Length 3:

1) BB ∪ {A, S }{A,C}
   BB ∪ AA ∪ AC ∪ SA ∪ SC
   =∅
2) **(A,C) (B)** ∪ B(A,C)
   AB ∪ CB ∪ BA ∪BC
   S → AB | BC   A → BA
   C → AB

Length 4:

B(S,A,C) ∪ (A,S) B ∪ ∅ **(A,C)**

= BS ∪ BA ∪ BC ∪ AB,SB

A → BA

S → BC

S → AB

C → AB

The string baaa belongs to the grammar