International Journal of Scientific Research Engineering & Technology (IJSRET)
Volume 1 Issue 5 pp 001-004 August 2012 www.ijsret.org ISSN 2278 – 0882

CYK Algorithm

Shamshad Ali
Computer Science & Engineering
Babu Banarasi Das Institute of Engineering Technology & Research Centre, Bulandshahr, U.P
shamshadali02@gmail.com

ABSTRACT

In this paper, we have to test the string that the string is a part of the given language. We have the substrings but we are not conformed that all these belong to given language. So how can we say that this string is that part of the given language or not. Now we are confused to determine it. To solve this problem we have a lot of way to check the membership of the given string. So that, we come to know about the membership of the sub-string or a string.

So to check the membership we have the algorithm that was invented by John Cocke, Taddo Kasami (in 1965) and Daniel H. Younger (in 1967). Although from the starting it was not so much popular. But now it is in used. It is time consuming. Because it takes too much time to solve and check the string whether it is the member or not. The reason behind this is the equation. Every step takes the help of equation. It is step by step process. If one step is calculated wrong then all the next will be wrong. So it takes more attention while computing.

It clearl-y checks the membership by the help of equation that value is given in the table. So we will test by the help of the table. If the starting symbol seems in the table entry $V_{1n.}$ then the string is the member of the language otherwise no. The whole table construction process takes $O(n^3)$ time in worst case. If w is of length **n** then there will be exactly **2n-1,** nodes labelled by variables in the tree.

Keywords: *Cocke-Younger-Kasami (CYK), Context free language (CFL), Chomsky Normal Form (CNF), Grammmar (G).*

I. INTRODUCTION

Basically the CYK algorithm is used to check or test the string whether the string belongs to the given language or not(i.e. the given string, is the member of the given language). We can describe membership of a string w in a **CFL**[1] L. There is an efficient technique based on the idea of "Dynamic Programming "which may known as "Table Filling Algorithm" or "Tabulation". This algorithm known as CYK Algorithm (**i.e. Cocke-Younger-Kasami**).

The algorithm works only if the grammar is in Chomsky normal form **(CNF)**[2] and succeeds by breaking one problem into a sequence of smaller one.
Assume that we have a CNF grammar
G = (V, T, P, S)[3] for a CFL language L and a string

$$w = a_1\ a_2\ a_3\ \ldots\ldots\ldots\ldots a_n \qquad \text{in } T^*$$

and the substrings are defined as

$$w_{ij} = a_i \ldots\ldots\ldots\ldots\ldots a_j$$

and the subsets that will be used in the table are defined as

$$V_{ij} = \{ A \in V : \ A =*\Rightarrow w_{ij} \}$$

It is simple that, $w \in L(G)$ if and only if $S \in V_{1n}$.
The algorithm constructs a table that tells whether w is in L. Note that when we will compute this w, the grammar itself is considered fixed, and its size contributes only a constant factor at the running time, which is measured in terms of the length of the string w whose membership in L is being tested.
In the CYK algorithm, we construct a triangular table as shown in the below figure 1.1

---

* CFL, CNF and G are defined at the last before the abbreviation used page.

Row 5	V_{15}				
Row 4	V_{14}	V_{25}			
Row 3	V_1	V_{24}	V_{35}		
Row 2	V_{12}	V_{23}	V_{34}	V_{45}	
Row 1	V_{11}	V_{22}	V_3	V_{44}	V_{55}
	a_1	a_2	a_3	a_4	a_5

Figure 1: CYK table.

The horizontal axis corresponds to the positions of the string, $w = a_1\ a_2\ a_3 \ldots\ldots\ldots\ldots a_n$, which we have

International Journal of Scientific Research Engineering & Technology (IJSRET)
Volume 1 Issue 5 pp 001-004 August 2012 www.ijsret.org ISSN 2278 – 0882

supposed has length 5. The table entry (or Elements of the Table) V_{ij} is the set of variables A such that:

$$A =*=> a_i\ a_{i+1}\ldots\ldots\ldots a_j\ .$$

To compute V_{ij}, observe that A ϵ V_{ij} if and only if G contains a production A→a_i . Therefore V_{ij} can be computed for all ($1 \leq i \leq n$) by inspection of w and the production of the grammar. To continue, notice that for j>I , A derives w_{ij} if and only if there is a production A→ BC, with B=*=> w_{ik} and C=*=> $w_{k+1,j}$ for some k with $i \leq k$, and k<j . In other words
V_{ij} = Ụ { A: A→ BC, with B ϵ V_{ik} and C ϵ $V_{k+1,j}$ }
k ϵ {i, i+1, …., j-1} .....Equation 1.1

Row 1, Row 2, ROW 3 …. are also computed as
Row 1: compute V_{11}, V_{22}, ……V_{nn}
Row 2: Compute V_{12}, V_{23},……$V_{n-1,n}$
Row 3: Compute V_{13}, V_{24},……$V_{n-2,n}$
and so on.

To fill the table, we work row by row, upwards. Notice that each row corresponds to one length of substrings; the bottom row is for strings of length1, the second from bottom row for strings of length 2, and so on until the top row corresponds to the one substring of length n, which is w itself. It takes O(n) time to compute any one entry of the table by a method. Since there are n(n+1)/2 table entries, the whole table construction process takes $O(n^3)$ time in worst case.
As we know Chomsky normal form (CNF) grammar are binary trees, if w is of length **n** then there will be exactly **2n-1,** nodes labeled by variables in the tree.
Example 1.1 The CNF grammar is defined a

S→ AB / BC A → BA / a
B → CC / b C → AB / a

Suppose w = baaba i.e. of length n = 5. If we will draw the binary tree, then there will be exactly 2n-1 = 2*5 -1 = 9 nodes labeled variables. Let us see by constructing tree

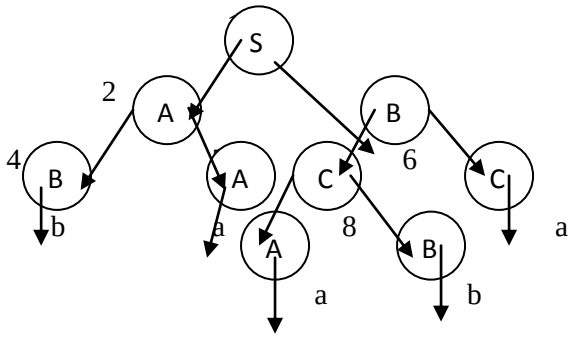

Figure 2: Binary tree

Here w= "baaba" i.e. length n=5.
Then no. of nodes must be 2n-1= 2*5-1=10-1=09. As shown in the figure there are 9 nodes variables.
In any condition if we draw binary tree either left most or right most of fixed length then there nodes always be as according to 2n-1.

Note: If we draw the table for this string w = baaba of length 5 then there will be total n (n+1)/ 2 entries will be found.

So, n (n+1)/2 = 5 (5+1)/2 = 15 entries.

V_{15}				
V_{14}	V_{25}			
V_{13}	V_{24}	V_{35}		
V_{12}	V_{23}	V_{34}	V_{45}	
V_{11}	V_{22}	V_{33}	V_{44}	V_{55}

Figure 3: CYK Simple Table

Total 15 entries and the values V_{ij}'s are computed by the equation 1.1.

II. ALGORITHM

Let the input be a string S consisting of n characters: a_1.....a_n.
Let the grammar contain r nonterminal symbols R_1.....Rr.
This grammar contains the subset Rs which is the set of start symbols.
Let P[n,n,r] be an array of Booleans. Initialize all elements of P to false.

for each i-1 to n
  for each unit production Rj → a_i
    set P[i,1,j] = true
for each i=2 to n .....length of span
  for each j=1 to n-i+1 .....Start of span
    for each k=1 to i-1 .....partition of span
      for each production RA → RB RC
        if P[j,k,B] and P[j+k, i-k, C]
        then set P[j,i,A]= true
if any of P[1,n,x] is true (x is iterated over the set s, where s are all the indices for Rs)
 then S is member of language
else
  S is not member of language.

Question 1: Use the CYK algorithm to determine whether the string w= baaba is in the language generated by the CNF grammar:

S→ AB / BC

A → BA / a
B → CC / b
C → AB / a

Solution:

STEP 1: We have to compute all the values of V_{ij} 's . That is Row 1 such as;

V_{11} is 'b' and 'b' is produced by B→b. So V_{11} = { B }, i.e. take the left side variable.

V_{22} is 'a' and 'a' is produced by A→a and C→a so we take union of A and C. so

V_{22} = { A, C }.Similarly

V_{11} = V_{44} = { B} and **V_{22} = V_{33} = V_{55} = { A, C }**

STEP 2: Now we will go upward that is we will compute Row2. We have to take the help of equation 1.1 (A→BC).

So, V_{12}→ V_{ik} $V_{k+1,j}$ here k=1 only so V_{ik} ϵ V_{11} and $V_{k+1,j}$ ϵ V_{22}. Put these values we find;

V_{12}→ V_{11} $V_{22,}$ Put the values of V_{11} and $V_{22.}$

➔ {B} {A,C}

➔ { BA, BC} BA and BC are produced by S→BC and A→BA

V_{12} → {S, A}.

Similarly,

V_{23} → V_{ik} $V_{k+1,j}$ here k=2 only so V_{ik} ϵ V_{22} and $V_{k+1,j}$ ϵ V_{33}

➔ V_{22} V_{33}

➔ {A,C} {A,C}

➔ { AA, AC, CA, CC }

AA, AC, CA, are not in the grammar.

But CC is in the grammar and produced by B→CC

So, **V_{23} → { B }.**

Similarly,

V_{34} → V_{ik} $V_{k+1,j}$ here k=3

so V_{ik} ϵ V_{33} and $V_{k+1,j}$ ϵ V_{44}

➔ V_{33} V_{44}

➔ {A,C} {B}

➔ { AB, CB}

CB is not in the grammar.

AB is in the grammar and produced by S→AB and C→AB

So, **V_{34} → {S,C}.**

Similarly,

V_{45} → V_{44} V_{55} here k=4

➔ {B} {A,C}

➔ { BA, BC}

BA is produced by A→BA and BC is produced by S→BC

V_{45} → {S,A}.

STEP 3: Now Row 3 will be computed

V_{13} → V_{ik} $V_{k+1,j}$ here k=1,2

So for k=1 V_{13} →V_{11} V_{23}

for k=2 V_{13} → V_{12} V_{33}

So we will take union of both. i.e.

V_{13} → V_{11} V_{23} U V_{12} V_{33}

➔ {B} {B} U {S,A} {A,C}

➔ {BB, SA, SC, AA, AC} Nothing are in the grammar so it is null.

V_{13} → {φ}

Similarly,

V_{24} → V_{22} V_{34} U V_{23} V_{44}

because here k=2 and 3.

➔ {A,C} {S,C} U {B} {B}

➔ {AS, AC, CS, CC, BB}

here only CC is produced by B → CC

So, **V_{24} → {B}**

Similarly,

V_{35} → V_{33} V_{45} U V_{34} V_{55}

here k=3 and 4.

➔ {A, C} {S,A} U {S,C}{A,C}

➔ { AS, AA, CS, CA, SA, SC, CA, CC }

here only CC is produced by B →CC

So, **V_{35} → {B}.**

STEP 4: Now we will compute Row 4 as above.

V_{14} → V_{ik} $V_{k+1,j}$ here k contains three values i.e. k= 1,2 and 3.

So for k=1 V_{14} → V_{11} V_{24}

For k=2 V_{14} → V_{12} V_{34}

For k=3 V_{14} → V_{13} V_{44}

V_{14} → V_{11} V_{24} U V_{12} V_{34} U V_{13} V_{44}

➔ {B}{B} U {S,A}{S,C} U{φ }{B}

➔ {BB, SS, SC, AS, AC, φ }

Nothing are produced.

So, **V_{14} → { φ }.**

Similarly,

V_{25} → V_{22} V_{35} U V_{23} V_{45} U V_{24} V_{55}

➔ {A,C}{B} U {B}{S,A} U {B}{A,C}

➔ {AB, CB, BS, BA, BC}

Here AB is produced by S→ AB and C→AB

BA is produced by A →BA

BC is produced by S→BC

So after taking union of left context i.e. A, S, and C.

V_{25} → { S,A,C }.

STEP 5: Now the last Row 5.

V_{15} → V_{ik} $V_{k+1,j}$ here the value of k is five i.e. k=1,2,3 and 4.

So for k=1 V_{15} → V_{11} V_{25}

for k=2 V_{15} → V_{12} V_{35}

for k=3 V_{15} → V_{13} V_{45}

for k=4 V_{15} → V_{14} V_{55}

V_{15}→ V_{11} V_{25} U V_{12} V_{35} U V_{13} V_{45} U V_{14} V_{55}

➔ {B}{S,A,C} U {S,A}{B} U φ U φ

→ {BS, BA, BC, SB, AB } BS and SB are not in the grammar.
BC is produced by S→BC
AB is produced by S→ AB and C→ AB
BA is produced by A → BA

So, V_{15} → **{S, A, C }.** Now put all the values in the table.

{S,A,C}				
{φ}	{S,A,C}			
{φ}	{B}	{B}		
{S,A}	{B}	{S,C}		{S,A}
{B}	{A,C}	{A,C}	{B}	{A,C}

Figure 4 Final CYK Table.

As we have got that the value of V_{15} is {S,A,C}. It contains 'S' which is the starting symbol. i.e. S ϵ V_{15}. So w= "baaba" is the member of the given grammar.

Exercise: Check out the membership for the string w="aabbb" by the CYK algo.
S →AB A → BB/a and B → AB/a.

CFL[1]: It is also called as Type-2 Production. The production is of the form
ά → ß where ά ϵ V and ß ϵ (V U Σ)*. For example, S → aSb, A →a/b, A→ ϵ etc. It is also called a context-free grammar. A language generated by a context-free grammar is called a context free language (CFL).

CNF[2]: In this form we have restrictions on the length of R.H.S. i.e. a Context free grammar G, is in CNF if every production is of the form
1.) A → BC, where A, B, C ϵ V.
i.e. One non-terminal → one non-terminal. One non-terminal
2.) A → a where A ϵ V and a ϵ T
One non-terminal → Terminal
Then this normal form is called CNF.

OR

For a grammar in CNF, the derivation tree has the following property: Every node has at most two descendents - either two internal vertices or a single leaf.

G[3]: Noam Chomsky gave the definition of a grammar. It is a simple 4-typle language i.e. G= (V, Σ, P, S), where
1.) **V** is a finite nonempty set whose elements are called Variables or Non-terminals. Mostly Capital letters are used. i.e. V={A,B,X,Y,……..}.
2.) **Σ** is a finite nonempty set whose elements are called Terminals. Mostly small letters are used. It is also denoted by '**T**'.
So **Σ** or T = {a,b,c,…x,y,…….}.
3.) V ∩ Σ = {φ}.
4.) **S** is a special variable (an element of V i.e. S ϵ V), or called start symbol.
5.) **P** is a finite set. The elements of p are called Productions.
i.e. P : S→ aS/ bS/a/b.

III. ABBREVIATION USED

CYK	:	Cocke-Younger-Kasami
CFL	:	Context-free language
CFG	:	Context -free grammar
CNF	:	Chomsky normal form
G	:	Grammar
V	:	Variable
P	:	Production
S	:	Special Symbol
T or Σ	:	Terminal
Vij	:	Table Entry
W	:	String
Wij	:	Substring
K	:	Integer
I	:	Integer
J	:	Integer

A,S,B,. All capital letters are Variables or non-terminals
a,b,c,… All small letters are Terminals.

ά	:	Left context
ß	:	Right context
n	:	Natural no.
$O(n^3)$	:	Worst case running time.
{φ}	:	Null
ϵ	:	Belongs to

REFERENCES

1. Hopcroft, Ullman, "Introduction to Automata Theory, Languages and Computation", Pearson Education.
2. K.L.P. Mishra and N. Chandrasekaran, "Theory of Computer Science: Automata, Languages and Computation", PHI
3. Peter Linz, "An Introduction to Formal Language and Automata", Narosa.
4. http://nitishkr.wordpress.com/2011/03/29/cyk-algorithm.implementation