The Cocke-Kasami-Younger algorithm

Given $w \in \Sigma^*$ & G a CFG

Want to know $w \in L(G)$?

We assume G is in Chomsky Normal Form. The parse trees are binary so for a string of length n we will have a tree with $(2n-1)$ variables. There are exponentially many such trees so we can generate them all, check if they are valid trees that generate w. Exponential time!

We can get this down to $O(n^3)$ using dynamic programming.

Given $G = (V, \Sigma, R, S)$

Input $w = a_1 \cdots a_n \in \Sigma^*$ $a_i \in \Sigma$

We work bottom up to construct a possible derivation of w. We first ask how we got each individual symbol.

Since G is in CNF we must have used rules of the form $A \to a$

We define inductively a 2-indexed family of subsets of V:

$X_{ij} := \{ A \in V \mid A \overset{*}{\Rightarrow} a_i \cdots a_j \}$

BASE CASE $X_{ii} = \{ A \in V \mid A \Rightarrow a_i \}$ $X_{11}, X_{22}, \cdots X_{nn}$

Next row will have $X_{12}, X_{23}, \cdots X_{i(i+1)} \cdots X_{(n-1)n}$

Next row will have $X_{13}, X_{24}, \cdots X_{i(i+2)} \cdots X_{(n-2)n}$

so X_{ij} will be in row $(j-i)+1$.

We compute and fill in the table ~~top~~ bottom to top.

When we compute X_{ij} we know X_{ik} & X_{kj} for all $i \le k \le j$

Now if $B \in X_{ik}$, $C \in X_{(k+1)j}$ & $A \to BC$ is a rule we know $A \in X_{ij}$. Why? $B \overset{*}{\Rightarrow} a_i \cdots a_k$ $C \overset{*}{\Rightarrow} a_{k+1} \cdots a_j$ so since $A \to BC \overset{*}{\Rightarrow} a_i \cdots a_j$.

$S \to AB \mid BC \qquad A \to BA \mid a \qquad B \to CC \mid b \qquad C \to AB \mid a$

We want to know $baaba \in L(G)$?

5	$\{A, S, C\}^{X_{15}}$				
4	$\phi^{X_{14}}$	$\{S, A, C\}^{X_{25}}$			
3	$\phi^{X_{13}}$	$\{B\}^{X_{24}}$	$\{B\}^{X_{35}}$		
2	$\{A, S\}$	$\{B\}$	$\{S, C\}$	$\{A, S\}$	
1	$\{B\}$	$\{A, C\}$	$\{A, C\}$	$\{B\}$	$\{A, C\}$
	b	a	a	b	a

We see that $S \in X_{15}$ so

$$S \overset{*}{\Rightarrow} a_1 \cdots a_5$$

In general $w \in L(G)$ iff $S \in X_{1n}$

Size of table $O(n^2)$

The time taken to find X_{ij} is $O(j-i)$

The time taken to compare X_{ik} & $X_{(k+1)j}$ & find a variable that generates them is $O(1)$: it depends on the size of the grammar but not on $\underline{n}$. So time to compute each X_{ij} is $O(n)$ & so overall $O(n^3)$.

There are better algorithms possible. Best is $O(n^{2.8})$.