CYK Algorithm

John Cocke 1970
Daniel Younger 1967
Tado Kasami 1965

Goal: Given a CFG G in Chomsky Normal Form, and a string w, determine if $w \in L(G)$.

We know that if $|w| = n$, it requires exactly $2n-1$ derivations.

Naive idea: Try out all derivations of $2n-1$ steps.
This is not time efficient.

CYK algorithm is based on dynamic programming.
This runs in $O(n^3)$ time.

$w_{24} = 111$
$w = 011100$

Break down the problem into similar sub problems.

Let $w = a_1 a_2 \ldots a_n$ where each $a_i \in \Sigma$.
Define $w_{ij} = a_i a_{i+1} \ldots a_j$ for all $1 \leq i \leq j \leq n$.

CYK builds T_{ij} for all $1 \le i \le j \le n$, such that

$$T_{ij} = \{ A \in V \mid A \overset{*}{\Rightarrow} w_{ij} \}$$

leading up to T_{1n}. Finally we check if the start variable $S \in T_{1n}$. That is, if $S \overset{*}{\Rightarrow} w_{1n} = w$.

$$\begin{array}{ccccccc}
T_{11} & T_{22} & T_{33} & \cdots & T_{nn} \\
T_{12} & T_{23} & T_{34} & \cdots & T_{n-1,n} \\
T_{13} & T_{24} & \cdots & T_{n-2,n} & \\
 & \vdots & & & \\
 & T_{1,n-1} & T_{2,n} & & \\
 & & T_{1,n} & &
\end{array}$$

Order of computing T_{ij}'s.

Algorithm outline

k+1 long substring

For $k=0$ to $n-1$, compute all $T(i, i+k)$

First we have $k=0$. $T_{i,i} = \{A \mid A \xRightarrow{*} w_{i,i} = a_i\}$

All the rules $A \to a_i$

For $k > 0$,

$w_{i,i+k} = \underbrace{a_i a_{i+1} \cdots a_{i+j}}_{w_{i,i+j}} \mid \underbrace{a_{i+j+1} \cdots a_{i+k}}_{w_{i+j+1,i+k}}$

$$A \in T_{i,i+k} \iff \begin{cases} B \in T_{i,i+j} \\ C \in T_{i+j+1,i+k} \\ A \to BC \text{ is a rule} \end{cases}$$

Algorithm

If $w = \varepsilon$, accept if $S \to \varepsilon$ is a rule

For $i=1$ to n

$A \in T_{ii} \iff A \to a_i$ is a rule

For $k=1$ to $n-1$

For $i=1$ to $n-k$

For $j=0$ to $k-1$

Check all the rules $A \to BC$

If $T_{i,i+j}$ contains B, and

$T_{i+j+1,k}$ contains C,

then $T_{i,i+k} = T_{i,i+k} \cup \{A\}$

$A \to B\ C$

$\underbrace{w_{i,i+j} \quad w_{i+j+1,i+k}}_{w_{i,i+k}}$

If $S \in T_{1,n}$, we say that $w \in L(G)$

Else $w \notin L(G)$.

Running Time: $O(n^3 r)$ where r is the number of rules. For a fixed grammar, r is considered to be a constant. $\to$ $O(n^3)$ algorithm

Correctness: Is evident from the algorithm.

$T_{i,j}$ – Contents of $T_{i,j}$

Example:

$S \to AB \mid BC$

$A \to BA \mid a$

$B \to CC \mid b$

$C \to AB \mid a$

$w = baaba$

j \ i	1	2	3	4	5
5	SAC	SAC	B	SA	AC
4	–	B	SC	B	
3	–	B	AC		
2	SA	AC			
1	B				

$w_{11} = b \quad T_{11} = \{B\}$

$w_{22} = a \quad T_{22} = \{A, C\}$

baaba

$T_{15} = \{S, A, C\}$

Since $S \in T_{15}$, we say that w is in $L(G)$.