The CYK Algorithm

The Cooke-Younger-Kasami (CYK) algorithm determines whether a particular string is in the language generated by a particular context-free grammar G, provide that G is in Chomsky Normal Form (CNF). CYK can be modified to work for any context-free grammar, however.

For the sake of simplicity of exposition, we will assume that G has no productions where the empty string is on the right hand side.

Our first step is to replace G by an equivalent grammar G' where the right hand side always has at most two symbols. We give an example below.

Suppose that the terminal alphabet of G is $\{a, b\}$, the only variable is the start symbol, S, and the productions are:

1. $S \to SS$
2. $S \to aSb$
3. $S \to ab$

(Note that G is ambiguous.)

We define G' to have the variables $\{S, A\}$ and the productions:

1. $S \to SS$
2. $S \to Ab$
3. $A \to aS$
4. $S \to ab$

Do you see how that was done?

We then define a dynamic program as follows, which, given a string w, computes one quadratic sized boolean array for each grammar symbol.

Let w be a string. If α is any grammar symbol (either a terminal or a variable) and $1 \leq i \leq j \leq |w|$, we define the boolean value $G_w[\alpha, i, j]$ to mean that α derives the substring of w consisting of the i^{th} through jth symbols in zero or more steps. If α is a terminal then $G_w[\alpha, i, j]$ is false if $i < j$, and is true if $i = j$ and the i^{th} symbol of w is α. If α is a variable, then $G_w[\alpha, i, j]$ is true if there is some production $\alpha \to \beta\gamma$ and some $i \leq k < j$ such that $G_w[\beta, i, k]$ and $G_w[\gamma, k+1, j]$, or if there is a production $\alpha \to \delta$ such that $G_w[i, j]$. Otherwise, $G_w[\alpha, i, j]$ is false.

The value of each $G_w[\alpha, i, j]$ can be computed in $O(j - i)$ time, by linear search through all possible values of k. Thus, the time complexity to compute all G is $O(n^3)$, where $n = |w|$.

G then generates w if and only if $G_w[S, 1, n]$.

We will now work through an example. Let $w = aabbabab$. Thus, $n = 8$. We draw a triangular array indexed by ordered pairs (i, j) where $1 \leq i \leq j \leq 8$. We draw the array as shown in Figure 1, because CYK

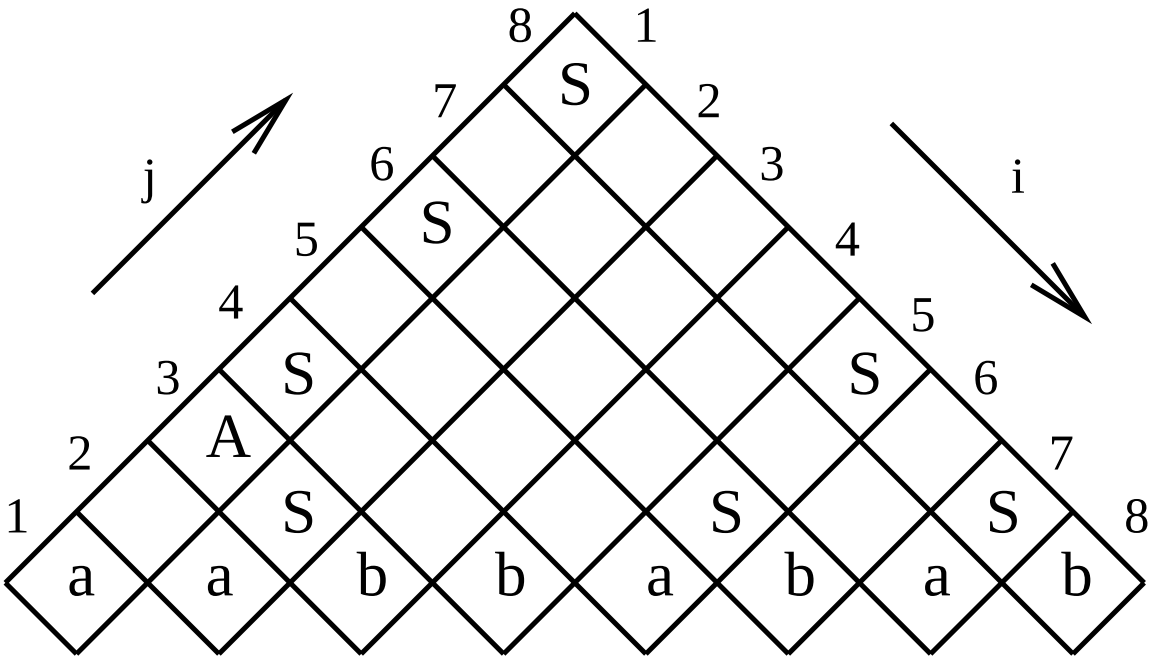


Figure 1: The CYK table filled in.

is a "bottom-up" algorithm, and it helps to put the abstract "bottom" subproblems literally at the bottom of the figure.

We then need to place a given grammar symbol α in a given cell (i, j) if and only if $G_w[\alpha, i, j]$. The bottom row is filled in immediately. We then fill in each row, ending at the top corner.

We know that G' generates w because there is an S in the top cell.

A cell could contain more than one symbol, although that does not occur in the above example.

Another Example

Consider the context-free grammar G:

1. $S \to a$
2. $S \to iS$
3. $S \to iSeS$

Replace G with an equivalent context-free grammar G' that has no more than two symbols on the right hand side of any production, as follows:

1. $S \to a$
2. $S \to iS$
3. $S \to IE$
4. $I \to iS$
5. $E \to eS$

Now, walk through the CYK algorithm for the string $w = iaeiiaea$, using Figure 2 below.

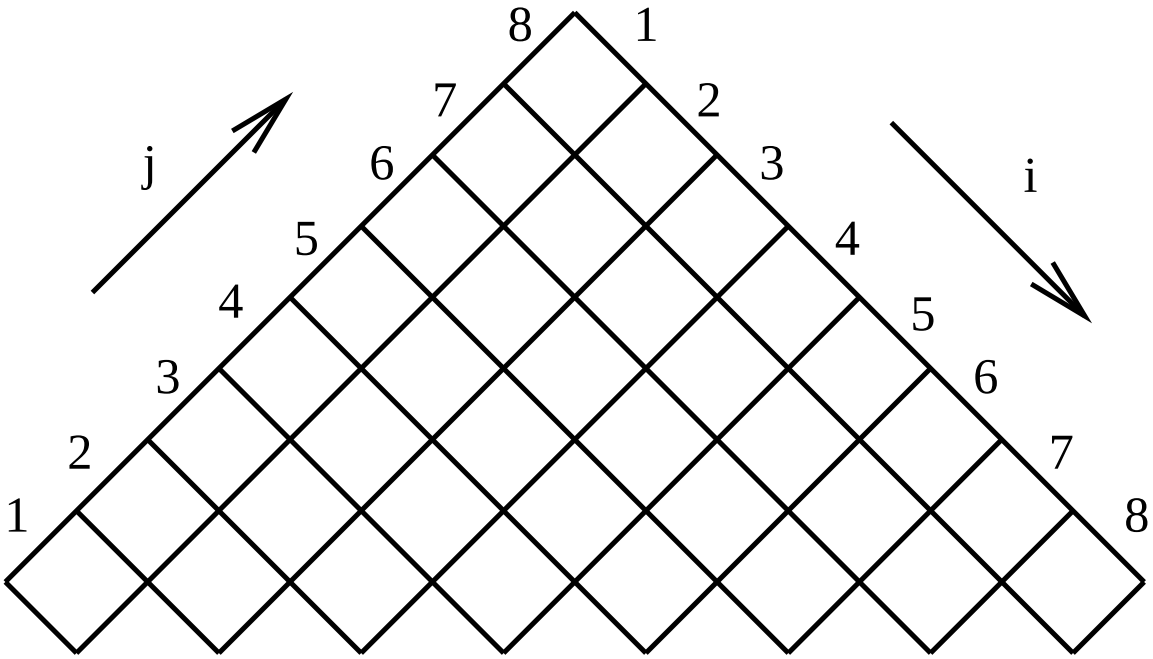


Figure 2: Fill in the cells, using the CYK algorithm.