The CYK Algorithm

Every Context-free language can be decided in polynomial time, using the CYK (Cook, Younger, and Kuratowski) dynamic programming algorithm.

Notation: If A is a variable of a context-free grammar with terminal alphabet Σ, we let $L(A)$ denote the set of strings over Σ can can be derived from A.

A Chomsky Normal Form grammar is a CF grammar with only two kinds of productions. The left-hand-side of one of these productions is, of course, a variable. The right-hand-side is either a terminal or two variables.

If L is any CFL which does not contain the empty string, there is a CNF grammar which generates L. If L is a CFL language which contains the empty string, we can simply delete the empty string, and the language is still context-free. Thus, there is some CNF grammar which generates $L - \{\lambda\}$. In this handout, we only consider languages which do not contain the empty string. The CYK algorithm determines whether a given string is a member of $L(G)$, where G is a Chomsky Normal Form grammar.

Subproblems of an Instance of CYK.

If $w = a_1a_2 \ldots a_n$ is any string, let $w[i, j]$ denote the substring $a_i \ldots a_j$, for any $1 \leq i \leq j \leq n$. An instance of the CYG membership problem is the ordered pair (G, w) where G is a context-free grammar and w is a string. That pair is a member of the CYG membership language if $w \in L(G)$.

A subproblem of that instance is a pair $(A, w[i, j])$, where A is a variable of the grammar G and $w[i, j]$ is a substring of w. The value of this subproblem is **true** if there is a derivation $A \stackrel{*}{\Rightarrow} w[i, j]$ using the grammar G, otherwise **false**. iff m is the number of variables of G, there are $m\binom{n+1}{2}$ subproblems instance.

Computing Subproblems

. Let G be a given CNF grammar and $w = a_1a_2 \ldots a_n$ a string. Let A be a variable of G.

1. For any $i \in \{1, \ldots n\}$ (A, i, i) is **true** if and only if $A \to a_i$ is a derivation of G.
2. For any $1 \leq i < j \leq n$, (A, i, j) is **true** if and only if, for variables B, C of G, $A \to BC$ is a derivation and (B, i, k) and $(C, k + 1, j)$ for some $i \leq k < j$.

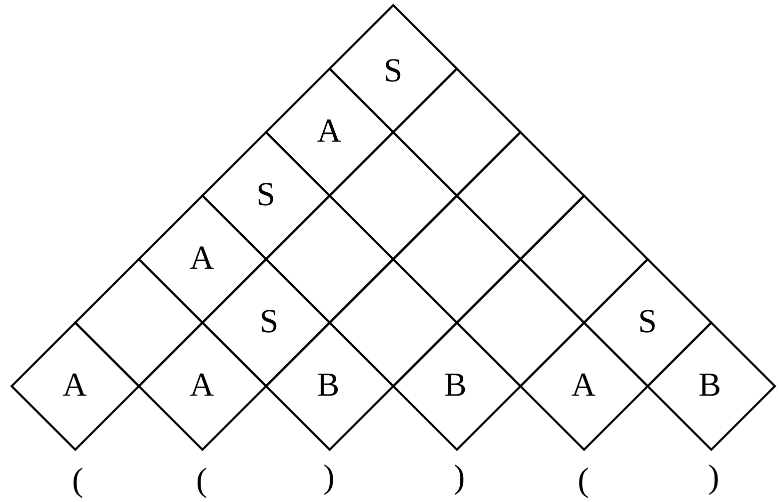
Walking Through CYK by Hand

The standard method of computing CYK by hand is to use a triangular matrix with $\binom{n+1}{2}$ entries, which we call *cells*, $C[i, j]$ for all $1 \leq i \leq j \leq n$. Each cell is drawn as a square, and the metrix consists of these $\binom{n+1}{2}$ squares. In descriptions on the internet, the matrix is drawn rectilinearly, but I find it more natural to place all $C[i, i]$ at the bottom level, all $C[i, i + 1]$ at the next level, and $C[1, n]$ at the top corner, each square at a 45^o angle. If A is a variable of G, then $A \in C[i, j]$ if and only if $(A, i, j) =$ **true**. Then $w \in L$ if and only if the start symbol is a member of $C[1, n]$.

Example: Dyck Language Let L be the Dyck language, minus the empty string. L is generated by the following CNF grammar.

$S \to AB$
$A \to ($
$B \to)$
$A \to AS$
$A \to SA$

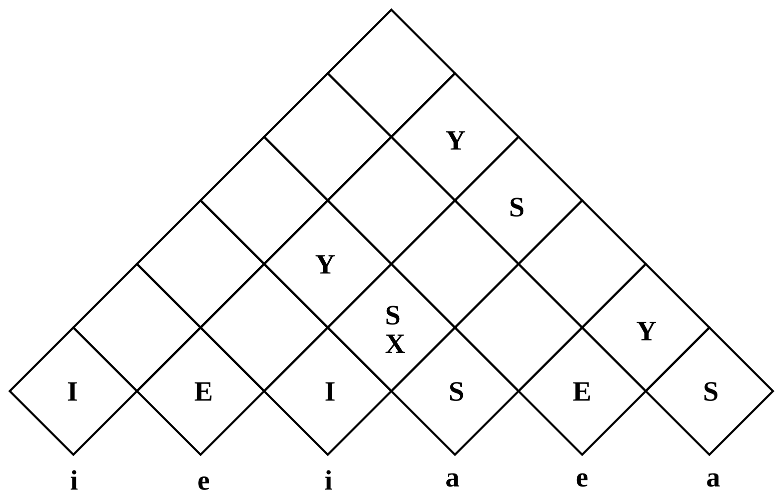
Let $w = (())()$, which we write below the figure. Filling in all the cells, we obtain the figure shown. $(())() \in L$, since $S \in C[1,6]$.



Example. Let G be the CNF grammar:

$S \to IS$
$S \to WS$
$S \to XY$
$X \to IS$
$Y \to ES$
$S \to a$
$E \to e$
$I \to i$
$W \to w$

Here is the CYK matrix with the initial string $w = ieiaea$ written below the bottom row. Since S is not in the top cell, $w \notin L$.



Indicating the Parse Tree. For the next example, we use the same grammar G as for the previous example. and we let $w = iiwaea$. G is ambiguous, and there are two parse trees for w, as shown in blue and red. Both parse trees can be found in the CYK matrix. A variable is in a cell because of one of the two computational rules given above. For example, $I \in C[2,2]$ because $I \to i$ and $a_2 = i$, and X is in $C[2,4]$ because $X \to IS$, $I \in C[2,2]$, and $S \in C[3,4]$.