INSTRUCTION MANUAL

CYK decision algorithm for CFG in CNF

By Juan Esteban Caicedo and Carlos Jimmy Pantoja

GENERATIVE GRAMMARS

A **generative grammar** is a quadruple, *G = (V, Σ, S, P)* formed by:

1. An alphabet *V* whose elements are called **variables** or **non-terminal symbols**.
2. An alphabet *Σ* whose elements are called **terminal symbols**. The alphabets *Σ* and *V* are required to be disjoint.
3. A special variable $S \in V$, called the **initial symbol** of the grammar.
4. A finite set $P \subseteq V \times (V \cup \Sigma)^*$ of **productions** or **rewriting rules**. A production $(A, w) \in P$ of *G* is denoted by $A \rightarrow w$ and is read "*A* produces *w*"; its meaning is: the variable *A* can be replaced (overwritten) by the string *w*. In the production $A \rightarrow w$, *A* is called the **head** and *w* the **body** of the production.

Variables are denoted by capital letters A, B, C, … The elements of *Σ* or terminal symbols are denoted by lowercase letters a, b, c, …

Starting with the initial symbol *S* and applying the productions of the grammar, in one or more steps, strings of terminals and/or non–terminals are obtained. Those strings that only have terminals make up what is called the *language generated by G*.

The **language generated by a grammar *G*** is defined as:

$$L(G) := \{w \in \Sigma^* : S \overset{+}{\Longrightarrow} w\}.$$

CONTEXT–FREE GRAMMARS

A **context–free grammar** (CFG), also called a **non–contextual grammar**, is a **type 2 generative grammar** in the *Chomsky hierarchy*.

A language L over an alphabet Σ is said to be a **context–free language** (CFL) if there exists a CFG G such that L(G) = L.

The name "context-free" comes from the fact that each production or rewriting rule $A \rightarrow w$ is applied to the variable A regardless of the characters that surround it, that is, regardless of the context in which A appears.

CHOMSKY NORMAL FORM

A GIC G is in **Chomsky Normal Form** (CNF) if it satisfies:

1. G has no useless variables.
2. G has no λ–productions (except possibly $S \rightarrow \lambda$).
3. All productions are of the form: $A \rightarrow a$ (simple productions) or $A \rightarrow BC$ (binary productions).

In particular, a grammar in CNF does not have unit productions.

Every GIC G *is equivalent to a Chomsky Normal Form grammar*.

CYK ALGORITHM

The CYK algorithm (name corresponding to the initials of researchers Cocke, Younger and Kasami) is a decision algorithm for CFG in CNF. This determines whether or not a given string of length n is generated by a CFG. The algorithm is applied by filling a table with n rows (one for each input terminal w) and n columns.

The pseudocode of the CYK algorithm is:

```
INPUT:  A grammar $G$ in CNF and a string of $n$ terminals $w = a_1a_2 \dots a_n$.
INITIALIZE:
    $j = 1$. For each $i$, $1 \le i \le n$,
    $X_{ij} = X_{i1}$ := set of variables $A$ such that $A \rightarrow a_i$ is a production of $G$.
REPEAT:
    $j := j + 1$. For each $i$, $1 \le i \le n - j + 1$,
    $X_{ij}$ := set of variables $A$ such that $A \rightarrow BC$ is a production of $G$, with $B \in X_{ik}$ and $C \in X_{i+k, j-k}$, considering all $k$ such that $1 \le k < j - 1$.
UNTIL:  $j = n$.
OUTPUT:  $w \in L(G)$ if and only if $S \in X_{1n}$.
```

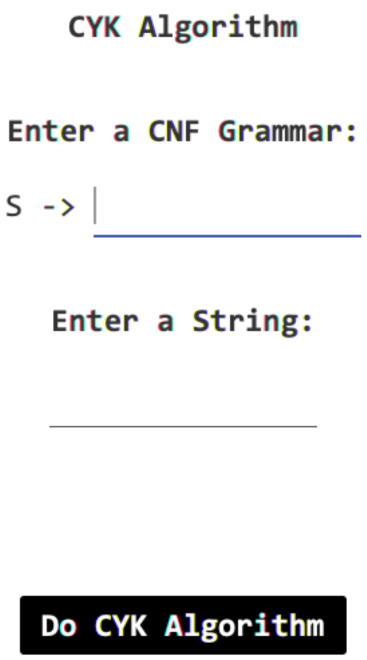
PROGRAM OPERATION

Step 1 – Enter a CNF grammar: The program only lets you write productions in the form $A \rightarrow a$ or $A \rightarrow BC$. In general, it always verifies that the grammar you write is in CNF.

The defined initial variable in this program is always S.

The defined “or” symbol for this program is ‘|’ and the one for the lambda symbol is ‘&’. Anyway, the program only lets you use these symbols for their corresponding objective.

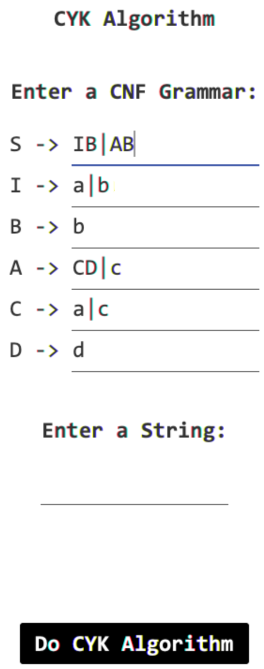
Every time you finish writing the productions of a variable, when you write new producer variables in the text field of that variable, new text fields of such variables will be automatically added in the GUI for you to write their corresponding productions. Just press "Enter" to make them appear or place the cursor in another text field.


CYK Algorithm

Enter a CNF Grammar:

S ->

Enter a String:

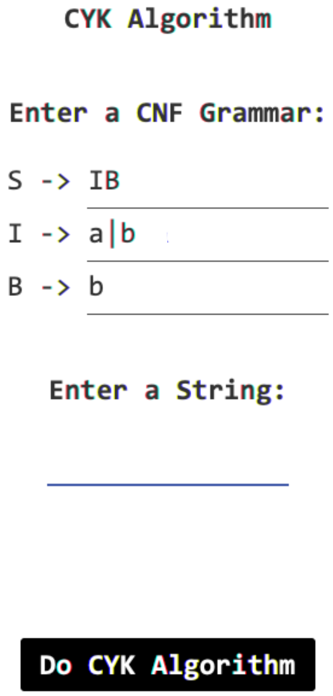

CYK Algorithm

Enter a CNF Grammar:

S -> IB|AB
I -> a|b
B -> b
A -> CD|c
C -> a|c
D -> d

Enter a String:

Similarly, if you completely remove the presence of a variable in the grammar, their corresponding productions will also be automatically deleted in cascade from the GUI, and therefore, the variables present in such productions will also be deleted with their corresponding productions.


CYK Algorithm

Enter a CNF Grammar:

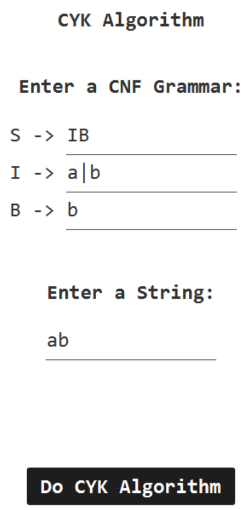
S -> IB
I -> a|b
B -> b

Enter a String:

Here we erased the production “AB” from the variable *S*.

Warning: The program does not let you place the cursor at the beginning or in the middle of the text of a field. So, if you want to erase whatever character you wrote, you will have to begin the erasion of the text from the end of the text field.

Step 2 – Enter a string: The program only lets you enter the alphabet symbols you wrote in the grammar. So, in the example below, we can just type the symbols 'a' and 'b' in the text field of the string.



CYK Algorithm

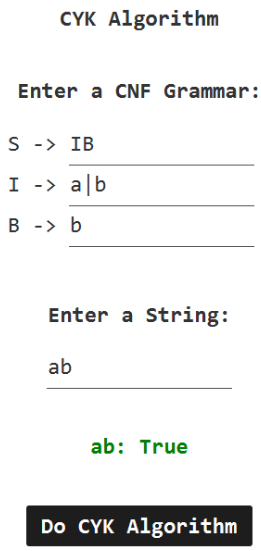
Enter a CNF Grammar:

```
S -> IB
I -> a|b
B -> b
```

Enter a String:

ab

Step 3 – Press the "Do CYK Algorithm" button: The result is Boolean. So, if the grammar generates the string, that is, if the string belongs to the grammar language, a label will appear in the form "string: True", otherwise in the form "string: False".



CYK Algorithm

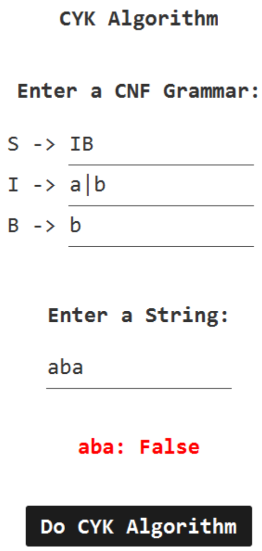
Enter a CNF Grammar:

```
S -> IB
I -> a|b
B -> b
```

Enter a String:

ab

ab: True



CYK Algorithm

Enter a CNF Grammar:

```
S -> IB
I -> a|b
B -> b
```

Enter a String:

aba

aba: False