Context Free Grammar

Table of Contents

[Parse Tree 3](#_Toc132659323)

[Converting DFA to CFG 6](#_Toc132659324)

[Chomsky Hierarchy 7](#_Toc132659325)

[Chomsky Normal Form 7](#_Toc132659326)

[Converting CFG to CNF 8](#_Toc132659327)

**Context Free Grammar** (CFG) is defined as . Here, is the set of variables, also called **non-terminal symbols**, is the set of alphabets, also called **terminal symbols**, is the set of rules, also called **production rules** and is the starting variable.

Since is the non-terminal symbols and is the terminal symbols, .

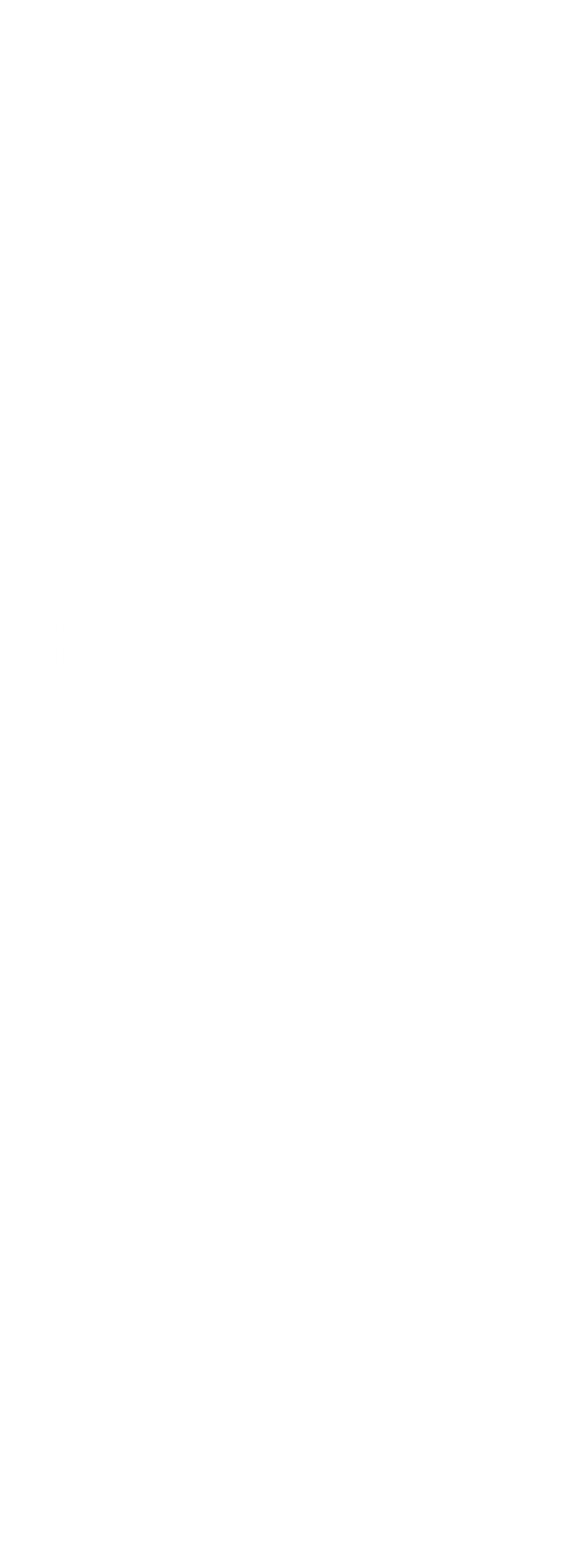
Suppose we have the production rule . Here, , and the starting variable, . This production can be used to generate a language, . The language generated by a context free grammar is called a **Context Free Language** (CFL).

The language as shown above is said to be in its **sentential form**. It can also be written in short form as .

Examples

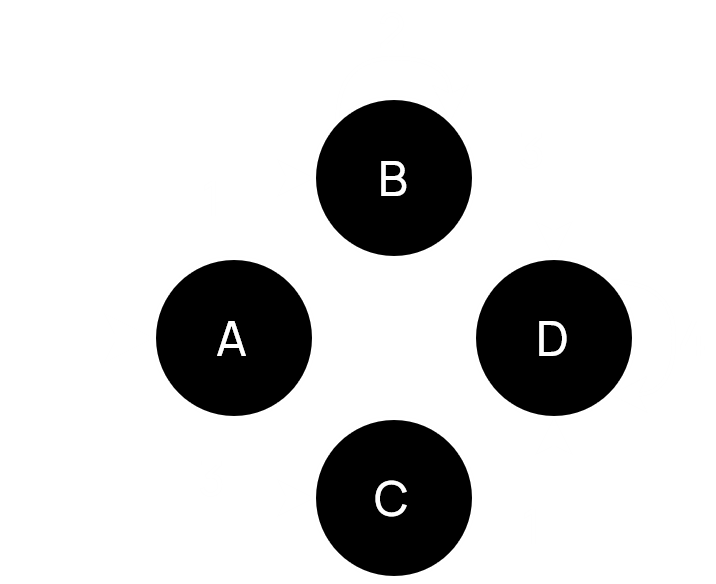
## Parse Tree

For the above equation, the **Parse Tree** looks like this:



It is not possible to generate a CFG for the above language, so it is not a CFL. In addition, it is also not a regular language since every regular language is a CFL.

## Converting DFA to CFG

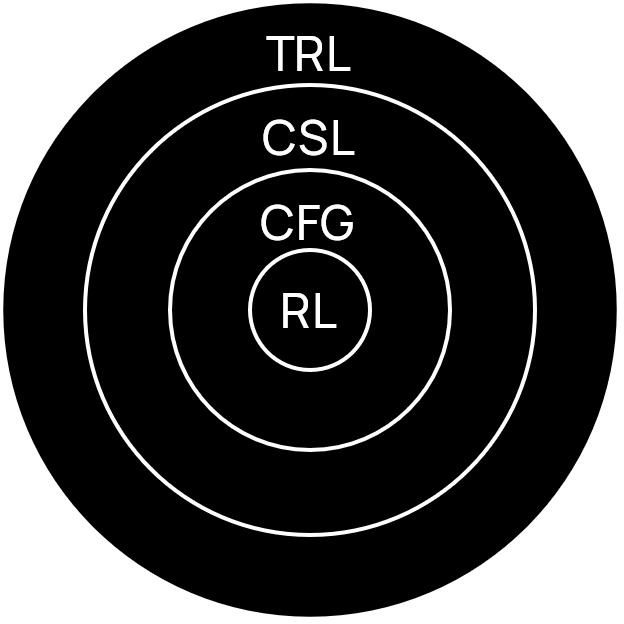


To convert a DFA to a CFG, we have to go through 4 steps:

1. Make a variable for each state.
2. Make a variable for the start state, i.e., the starting variable.
3. Make a rule for each edge.
4. Add a rule for the accepting state.

For the DFA shown above, the rules are:

## Chomsky Hierarchy



The **Chomsky Hierarchy** is a hierarchy of language categories. In the deepest circle, we have Type 3 languages, which are **regular languages**. In the circle outside that we have Type 2 languages, **context free languages**. Outside that are Type 1 languages, which are called **concept sensitive languages**. On the outer-most circle, we have Type 0 languages, known as **Turing recognizable languages**.

## Chomsky Normal Form

The **Chomsky Normal Form** (CNF) is a form for writing **production rules**. It has the format or . Thus, it is possible to write the rule as a combination of exactly 2 variables, neither of which are the starting variable, or as a single terminal state. There is one exception to this form just for starting variables, .

## Converting CFG to CNF

Every CFG can be represented as a CNF. To do this we must follow the following rules:

1. Make sure the starting symbol does not appear on the right-hand side of any production rules.
2. Remove production rules, e.g., .
3. Get rid of all unit production rules, e.g., .
4. Get rid of all rules with more than 2 symbols on the right-hand side, e.g., .
5. All rules must be in the form or .

Consider the following set of production rules:

These do not break rule 1, but they do break rule 2. We must remove and . We do this by placing anywhere appears, and anywhere appears.

This setup still violates rule 3. To get rid of and , we must replace them with and respectively.

Finally, we have combinations of variables and terminal states, which we are not allowed to have. To replace those, we need to introduce new variables. Let , and .