# **Chomsky's Normal Form (CNF)**

CNF stands for Chomsky normal form. A CFG(context free grammar) is in CNF(Chomsky normal form) if all production rules satisfy one of the following conditions:

* Start symbol generating ε. For example, A → ε.
* A non-terminal generating two non-terminals. For example, S → AB.
* A non-terminal generating a terminal. For example, S → a.

### **For example:**

1. G1 = {S → AB, S → c, A → a, B → b}
2. G2 = {S → aA, A → a, B → c}

The production rules of Grammar G1 satisfy the rules specified for CNF, so the grammar G1 is in CNF. However, the production rule of Grammar G2 does not satisfy the rules specified for CNF as S → aZ contains terminal followed by non-terminal. So the grammar G2 is not in CNF.

## **Steps for converting CFG into CNF**

**Step 1:** Eliminate start symbol from the RHS. If the start symbol T is at the right-hand side of any production, create a new production as:

1. S1 → S

Where S1 is the new start symbol.

**Step 2:** In the grammar, remove the null, unit and useless productions. You can refer to the [Simplification of CFG](https://www.javatpoint.com/automata-simplification-of-cfg).

**Step 3:** Eliminate terminals from the RHS of the production if they exist with other non-terminals or terminals. For example, production S → aA can be decomposed as:

1. S → RA
2. R → a

**Step 4:** Eliminate RHS with more than two non-terminals. For example, S → ASB can be decomposed as:

1. S → RS
2. R → AS

### **Example:**

Convert the given CFG to CNF. Consider the given grammar G1:

1. S → a | aA | B
2. A → aBB | ε
3. B → Aa | b

**Solution:**

**Step 1:** We will create a new production S1 → S, as the start symbol S appears on the RHS. The grammar will be:

1. S1 → S
2. S → a | aA | B
3. A → aBB | ε
4. B → Aa | b

**Step 2:** As grammar G1 contains A → ε null production, its removal from the grammar yields:

1. S1 → S
2. S → a | aA | B
3. A → aBB
4. B → Aa | b | a

Now, as grammar G1 contains Unit production S → B, its removal yield:

1. S1 → S
2. S → a | aA | Aa | b
3. A → aBB
4. B → Aa | b | a

Also remove the unit production S1 → S, its removal from the grammar yields:

1. S0 → a | aA | Aa | b
2. S → a | aA | Aa | b
3. A → aBB
4. B → Aa | b | a

**Step 3:** In the production rule S0 → aA | Aa, S → aA | Aa, A → aBB and B → Aa, terminal a exists on RHS with non-terminals. So we will replace terminal a with X:

1. S0 → a | XA | AX | b
2. S → a | XA | AX | b
3. A → XBB
4. B → AX | b | a
5. X → a

**Step 4:** In the production rule A → XBB, RHS has more than two symbols, removing it from grammar yield:

1. S0 → a | XA | AX | b
2. S → a | XA | AX | b
3. A → RB
4. B → AX | b | a
5. X → a
6. R → XB

Hence, for the given grammar, this is the required CNF.

# **Greibach Normal Form (GNF)**

GNF stands for Greibach normal form. A CFG(context free grammar) is in GNF(Greibach normal form) if all the production rules satisfy one of the following conditions:

* A start symbol generating ε. For example, S → ε.
* A non-terminal generating a terminal. For example, A → a.
* A non-terminal generating a terminal which is followed by any number of non-terminals. For example, S → aASB.

**For example:**

The production rules of Grammar G1 satisfy the rules specified for GNF, so the grammar G1 is in GNF. However, the production rule of Grammar G2 does not satisfy the rules specified for GNF as A → ε and B → ε contains ε(only start symbol can generate ε). So the grammar G2 is not in GNF.

## **Steps for converting CFG into GNF**

**Step 1:** Convert the grammar into CNF.

If the given grammar is not in CNF, convert it into CNF. You can refer the following topic to convert the CFG into CNF: Chomsky normal form

**Step 2:** If the grammar exists left recursion, eliminate it.

If the context free grammar contains left recursion, eliminate it. You can refer the following topic to eliminate left recursion: Left Recursion

**Step 3:** In the grammar, convert the given production rule into GNF form.

If any production rule in the grammar is not in GNF form, convert it.

### **Example:**

**Solution:**

As the given grammar G is already in CNF and there is no left recursion, so we can skip step 1 and step 2 and directly go to step 3.

The production rule A → SA is not in GNF, so we substitute S → XB | AA in the production rule A → SA as:

The production rule S → XB and B → XBA is not in GNF, so we substitute X → a in the production rule S → XB and B → XBA as:

Now we will remove left recursion (A → AAA), we get:

Now we will remove null production C → ε, we get:

The production rule S → AA is not in GNF, so we substitute A → aC | aBAC | a | aBA in production rule S → AA as:

The production rule C → AAC is not in GNF, so we substitute A → aC | aBAC | a | aBA in production rule C → AAC as:

Hence, this is the GNF form for the grammar G.