Simplification of Context-free Grammars

Raju Hazari

Department of Computer Science and Engineering
National Institute of Technology Calicut

November 14, 2023

Simplification of CFG

- When you want to write a programming language, you have to write a complier and first step is to write a grammar.
- Without knowing it properly, you may end up with some useless symbols, because when you write a grammar for a language it will not have 3 or 4 rules, it will have hundres of rules.
- So, when you write hundres of rules, without knowing it properly there may be some useless symbols which will not lead you to anything.

Simplification of CFG

- When you want to write a programming language, you have to write a complier and first step is to write a grammar.
- Without knowing it properly, you may end up with some useless symbols, because when you write a grammar for a language it will not have 3 or 4 rules, it will have hundres of rules.
- So, when you write hundres of rules, without knowing it properly there may be some useless symbols which will not lead you to anything.
- So, how to find out which symbols are useless and how to get rid of them ?

Useless symbols and productions

- **Definition 1:** Let $G = (V, T, P, S)$ be a context-free grammar. A symbol or variable $A \in V$ is said to be **useful** if and only if there is at least one $w \in L(G)$ such that
$$S \overset{*}{\Rightarrow} xAy \overset{*}{\Rightarrow} w,$$
with x, y in $(V \cup T)^*$.
    - In words, a variable is useful if and only if it occurs in at least one derivation.
    - A variable that is not useful is called **useless**.
    - A **production is useless** if it involves any useless variable.

Useless symbols and productions

- **Example 1:** Let us consider the grammar G whose entire production set is:

$$S \to aSb,$$
$$S \to ab,$$
$$S \to A,$$
$$A \to aA$$

  - the production $S \to A$ clearly plays no role, as A cannot transformed into a terminal string.
  - While A can occur in a string derived from S, this can never lead to a sentence.
  - Removing this production leaves the language unaffected.
  - Here, A is **useless symbol** and productions $S \to A$, $A \to aA$ are **useless production**.

Useless symbols and productions

- **Example 2:** Let us consider the grammar G whose entire production set is:

$$\begin{aligned} &S \rightarrow A, \\ &A \rightarrow aA, \\ &A \rightarrow a, \\ &B \rightarrow bA, \\ &B \rightarrow b \end{aligned}$$

  - the variable B is useless and so is the productions $B \rightarrow bA$, $B \rightarrow b$.
  - Although B can drive a terminal string, there is no way we can achieve $S \overset{*}{\Rightarrow} xBy$.

Useless symbols and productions

- If you look carefully at the **Definition 1**, there are two aspects :
    1. It should be possible to derive a terminal string from A.
    2. It should be possible to derive a sentential form containing A from S. (or A is reachable from S)
- There are two conditions, these two conditions are necessary condition but they are not sufficient.

Useless symbols and productions

- If you look carefully at the **Definition 1**, there are two aspects :
    1. It should be possible to derive a terminal string from A.
    2. It should be possible to derive a sentential form containing A from S. (or A is reachable from S)
- There are two conditions, these two conditions are necessary condition but they are not sufficient. Why ?

Useless symbols and productions

Let us consider the productions:

$$\begin{aligned} S &\rightarrow aABbc, \\ A &\rightarrow ac, \\ S &\rightarrow cDd, \\ D &\rightarrow cDd, \\ D &\rightarrow cd \end{aligned}$$

Now, is the symbol A useful or not ?

Useless symbols and productions

Let us consider the productions:

$$\begin{aligned} S &\rightarrow aABbc, \\ A &\rightarrow ac, \\ S &\rightarrow cDd, \\ D &\rightarrow cDd, \\ D &\rightarrow cd \end{aligned}$$

Now, is the symbol A useful or not ?

- It is possible to get a sentential form containing A from S.
- It is also possible to derive a terminal string from A.
- But, if you use the production $S \rightarrow aABbc$ and replace A by ac, still B will be there and there is no rule with B on the left hand side. It is not possible to derive a terminal string from B.
- So, if a symbol is useful, then it should also be possible to derive a terminal string from X, and it should also possible to derive a sentential form containing X from S, but these two conditions in put together will not imply the symbol is useful.
- So, they are necessary conditions, but they are not sufficient conditions.

Removing Useless Productions

- **Lemma 1:** Given a CFG $G = (V, T, P, S)$, with $L(G) \neq \Phi$. We can efficiently find an equivalent CFG $G^{'} = (V^{'}, T, P^{'}, S)$, such that for each A in $V^{'}$ there is some w in T^* for which $A \Rightarrow w$.
  - Given a grammar $G = (V, T, P, S)$, you construct a grammar $G^{'} = (V^{'}, T, P^{'}, S)$
  - The language generates for both are the same, it is an equivalent grammar.
  - But, in $G^{'}$, any non-terminal A belonging to $V^{'}$ will have the property that it should be possible to derive a terminal string w from A.
  - In this case the terminals are not really affected, so we use the same T. But, $V^{'}$ will be a subset of V, and $P^{'}$ will be a subset of P. You just remove some non-terminals and productions involving them, so that the resultant grammar.

Removing Useless Productions

- The steps are very simple. We use this algorithm to find the set of non-terminals

```
begin
1) $OLD\ V := \Phi$
2) $NEW\ V := \{A \mid A \to w$ for some $w$ in $T^*\}$;
3) while $OLD\ V \neq NEW\ V$ do
        begin
4)      $OLD\ V := NEW\ V$ ;
5)      $NEW\ V := OLD\ V \cup \{A \mid A \to \alpha$ for some $\alpha$ in $(T \cup OLD\ V)^*\}$
        end
6) $V' := NEW\ V$
end
```

Removing Useless Productions

- Initially, we look at rules of the forms $A \rightarrow ab$.
- Look at the production in P and look at symbols where the right hand side is a terminal one, put those non-terminals in $NEW\ V$.
- Then look at those rules, where the right hand side will be consist of terminal symbols and symbols already present in $NEW\ V$, include that in $NEW\ V$
- Keep on doing that until no more things can be added.

Removing Useless Productions

- So, let us take a very simple example and see the idea

$$\begin{aligned}
&S \to AB,\\
&S \to AC,\\
&B \to b,\\
&S \to BC,\\
&C \to cEd,\\
&E \to cEd,\\
&E \to cd,\\
&A \to aA,
\end{aligned}$$

Removing Useless Productions

- So, let us take a very simple example and see the idea

$$
\begin{aligned}
&S \rightarrow AB,\\
&S \rightarrow AC,\\
&B \rightarrow b,\\
&S \rightarrow BC,\\
&C \rightarrow cEd,\\
&E \rightarrow cEd,\\
&E \rightarrow cd,\\
&A \rightarrow aA,
\end{aligned}
$$

V = S, A, B, C, E

Removing Useless Productions

- So, let us take a very simple example and see the idea

$$\begin{aligned}
&S \to AB,\\
&S \to AC,\\
&B \to b,\\
&S \to BC,\\
&C \to cEd,\\
&E \to cEd,\\
&E \to cd,\\
&A \to aA,
\end{aligned}$$

V = S, A, B, C, E

NEW V = {B, E}

Removing Useless Productions

- So, let us take a very simple example and see the idea

$$\begin{aligned}
&S \to AB,\\
&S \to AC,\\
&B \to b,\\
&S \to BC,\\
&C \to cEd,\\
&E \to cEd,\\
&E \to cd,\\
&A \to aA,
\end{aligned}$$

V = S, A, B, C, E

NEW V = {B, E} OLD V = {B, E}

Removing Useless Productions

- So, let us take a very simple example and see the idea

$$\begin{aligned}
&S \rightarrow AB,\\
&S \rightarrow AC,\\
&B \rightarrow b,\\
&S \rightarrow BC,\\
&C \rightarrow cEd,\\
&E \rightarrow cEd,\\
&E \rightarrow cd,\\
&A \rightarrow aA,
\end{aligned}$$

V = S, A, B, C, E

NEW V = {B, E} OLD V = {B, E}

NEW V = {B, E, C}

Removing Useless Productions

- So, let us take a very simple example and see the idea

$$
\begin{aligned}
&S \rightarrow AB, \\
&S \rightarrow AC, \\
&B \rightarrow b, \\
&S \rightarrow BC, \\
&C \rightarrow cEd, \\
&E \rightarrow cEd, \\
&E \rightarrow cd, \\
&A \rightarrow aA,
\end{aligned}
$$

V = S, A, B, C, E

NEW V = {B, E}	OLD V = {B, E}
NEW V = {B, E, C}	OLD V = {B, E, C}

Removing Useless Productions

- So, let us take a very simple example and see the idea

$$
\begin{aligned}
&S \to AB,\\
&S \to AC,\\
&B \to b,\\
&S \to BC,\\
&C \to cEd,\\
&E \to cEd,\\
&E \to cd,\\
&A \to aA,
\end{aligned}
$$

V = S, A, B, C, E

NEW V = {B, E}	OLD V = {B, E}
NEW V = {B, E, C}	OLD V = {B, E, C}
NEW V = {B, E, C, S}	

Removing Useless Productions

- So, let us take a very simple example and see the idea

$$
\begin{aligned}
&S \rightarrow AB,\\
&S \rightarrow AC,\\
&B \rightarrow b,\\
&S \rightarrow BC,\\
&C \rightarrow cEd,\\
&E \rightarrow cEd,\\
&E \rightarrow cd,\\
&A \rightarrow aA,
\end{aligned}
$$

V = S, A, B, C, E

NEW V = {B, E}	OLD V = {B, E}
NEW V = {B, E, C}	OLD V = {B, E, C}
NEW V = {B, E, C, S}	OLD V = {B, E, C, S}

Removing Useless Productions

- So, let us take a very simple example and see the idea

$$
\begin{aligned}
&S \to AB,\\
&S \to AC,\\
&B \to b,\\
&S \to BC,\\
&C \to cEd,\\
&E \to cEd,\\
&E \to cd,\\
&A \to aA,
\end{aligned}
$$

V = S, A, B, C, E

NEW V = {B, E}	OLD V = {B, E}
NEW V = {B, E, C}	OLD V = {B, E, C}
NEW V = {B, E, C, S}	OLD V = {B, E, C, S}

A is useless symbols. So, remove A and the rules involving A on the left or on the right. So, remove $S \to AB$, $S \to AC$, $A \to aA$.

Removing Useless Productions

- **Lemma 2:** Given a CFG $G = (V, T, P, S)$ we can effectively find an equivalent CFG $G^{'} = (V^{'}, T^{'}, P^{'}, S)$ such that for each A in $V^{'} \cup T^{'}$ there exists x and $y \in (V^{'} \cup T^{'})^*$ for which $S \Rightarrow xAy$.
    - Given a CFG $G = (V, T, P, S)$, now from this we want to construct another grammar $G^{'} = (V^{'}, T^{'}, P^{'}, S)$ where the non-terminals $(V^{'})$ will be subset of V, terminals $(T^{'})$ will be a subset of T, $P^{'}$ will be a subset of P.
    - Such that from S it should be possible to get a sentential form in which each of them are correct.

Removing Useless Productions

- Let us consider the following grammar :

$$\begin{array}{ll} S \to AB, & B \to b, \\ A \to aAb, & C \to cCd, \\ A \to ab, & C \to cd \\ B \to bB, & \end{array}$$

  - Now, $V = \{S, A, B, C\}$ and $T = \{a, b, c, d\}$
  - Now start with V' and T', first put S in V', then look at the rules with S on the left hand side, and look at the terminal symbols and non-terminals symbols occurring on the right. $V' = \{S\}$
  - Here, $S \to AB$. So, A, B are non-terminals you include then in V'. $V' = \{S, A, B\}$
  - Then look at the rule with A or B on the left hand side. On the right hand side we get non-terminal symbols A and B and, terminal symbols a and b. So, we will adding a and b in T'. $T' = \{a, b\}$
  - There are no more non-terminals added, so no more rules you need to consider.
  - C is not reachable from S, so from S it is not possible to get a sentential form containing C. So, C is a useless non-terminal and c, d are useless terminals, and $C \to cCd$ and $C \to cd$ are useless production rules.

Removing Useless Productions

- **Theorem 1:** Every non-empty CFL is generating by a CFG with no useless symbols.
    - So, every nonempty context-free language, we are generated by a context free grammar with no useless symbols.

Removing Useless Productions

- **Theorem 1:** Every non-empty CFL is generating by a CFG with no useless symbols.
  - So, every nonempty context-free language, we are generated by a context free grammar with no useless symbols. How we do that ?

Removing Useless Productions

- **Theorem 1:** Every non-empty CFL is generating by a CFG with no useless symbols.
  - So, every nonempty context-free language, we are generated by a context free grammar with no useless symbols. How we do that ?

$$\begin{tikzcd}
G = (V, T, P, S) \arrow[d, Rightarrow, "\text{apply Lemma 1}"] \\
G' = (V', T, P', S) \arrow[d, Rightarrow, "\text{apply Lemma 2}"] \\
G'' = (V'', T'', P'', S)
\end{tikzcd}$$

Removing Useless Productions

- **Theorem 1:** Every non-empty CFL is generating by a CFG with no useless symbols.
  - So, every nonempty context-free language, we are generated by a context free grammar with no useless symbols. How we do that ?

$$\begin{tikzcd}
G = (V, T, P, S) \arrow[d, Rightarrow, "\text{apply Lemma 1}"] \\
G' = (V', T, P', S) \arrow[d, Rightarrow, "\text{apply Lemma 2}"] \\
G'' = (V'', T'', P'', S)
\end{tikzcd}$$

  - $V^{'}$ is subset of V, $V^{''}$ is subset of $V^{'}$. So, by transitivity, $V^{''}$ will be subset of V.
  - G and $G^{'}$ the T are equal but between $G^{'}$ and $G^{''}$, $T^{''}$ is subset of T.
  - $P^{'}$ is subset of P, $P^{''}$ is subset of $P^{'}$.

Removing Useless Productions

- **Claim :** We claim that G'' has no useless symbols.

Removing Useless Productions

- **Claim :** We claim that G'' has no useless symbols.
- **Proof :** Suppose G'' has a useless symbol X. Now, G'' is obtained from G' by applying lemma 2. So, every symbol here is reachable from S. G'' has that property. So, from S it should be possible to get something like $S \overset{*}{\Rightarrow} \alpha X \beta$. Now, every symbol here either non-terminal or terminal, especially non-terminals, whatever non-terminals are occurring in α or β they are in V', because V'' is subset of V'. So, whatever non-terminals is appearing in $S \overset{*}{\Rightarrow} \alpha X \beta$ that will be in V' and G' has the property that every non-terminals derives a terminal string. So, from every symbol in $S \overset{*}{\Rightarrow} \alpha X \beta$ it should be possible to drive a terminal string. So, we are able to get a derivation like $S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$, which means that X is useful. So, the assumption is not correct. So, G'' do not have useless symbols.

Removing Useless Productions

- Now, staring from G, we have apply Lemma 1 to get $G^{'}$, and we have apply Lemma 2 to get $G^{''}$.
- These lemmas should be applied in that order only. You can not apply Lemma 2 first and then Lemma 1.
- If you change the order, you may not able to remove all the useless productions.
- For example, consider the grammar
$$S \rightarrow AB,$$
$$S \rightarrow a,$$
$$A \rightarrow a$$
- The language consists of only one string a. $L(G) = \{a\}$. So, it is enough if we have the rule $S \rightarrow a$.
- If we apply Lemma 2 and then Lemma 1, then we will end up with a grammar $S \rightarrow a$, $A \rightarrow a$, but $A \rightarrow a$ is useless.
- If we apply Lemma 1 and then Lemma 2, then we will end up with a grammar $S \rightarrow a$. Only one rule.

ϵ-Production and Unit Production

- **Definition :** ϵ-Production:
  Any production of a context-free grammar of the form $A \to \epsilon$, is called an **ϵ-production**.

- **Definition :** Unit Production:
  Any production of a context free grammar of the form $A \to B$, where $A, B \in V$, is called a **unit production**.

ϵ-Production and Unit Production

- **Definition :** ϵ-Production:
  Any production of a context-free grammar of the form $A \to \epsilon$, is called an **ϵ-production**.
- **Definition :** Unit Production:
  Any production of a context free grammar of the form $A \to B$, where $A, B \in V$, is called a **unit production**.
- Why we avoid such productions ?

ϵ-Production and Unit Production

- **Definition :** ϵ-Production:
  Any production of a context-free grammar of the form $A \to \epsilon$, is called an **ϵ-production**.

- **Definition :** Unit Production:
  Any production of a context free grammar of the form $A \to B$, where $A, B \in V$, is called a **unit production**.

- Why we avoid such productions ?
  - First of all that length increasing property, that length of the left hand side is less than or equal to length of the right hand side. It is violated by the rule in the form $A \to \epsilon$. So, in proofs when this property violated, the proofs becomes lengthy. So, if we avoid such rules the proof we can give very easily.

ϵ-Production and Unit Production

- Why we avoid such productions ? Contd...
  - In compiler every time when the parsing takes place, the reduction either bottom up or top down it is taking place. So, when you reduce using a rule, a semantic routine will be called and a code will be generated. So, if these unit production really do not do anything or these unit productions really do not contribute to any proper code but unnecessarily we will be using them and each time you will be calling a semantic routine. So, if we use too many of that unit productions $A \to B$, the compiler time will be more.
  - When we avoid the unit productions the number of rules may increase, but the number of rules increase very much then also it is a disadvantage. At each stage we may have to check which rule is applicable for reduction. if there are two many choices then that time will be more. So, the idea is to reduce the compiler time by removing the unit production.

Removing ϵ-Productions

- If we avoid ϵ-Production the language generated will not have ϵ. If we want to have ϵ, we must have atleast one rule $S \rightarrow \epsilon$. If we add the rule $S \rightarrow \epsilon$, we must also make sure that S does not occur on the right hand side of the any production.
- At this stage we will assume that $L(G)$ does not contain ϵ.
- **Theorem :** If $L = L(G)$ for some CFG $G = (V, T, P, S)$, then $L - \{\epsilon\}$ is $L(G^{'})$ for a CFG $G^{'}$ with no useless symbols or ϵ-production.
  - We assume that L does not contain ϵ and we have to show that $L = L(G^{'})$ and $G^{'}$ does not have ϵ-production.
  - Let $G = (N, T, P, S)$ then $G^{'} = (N, T, P^{'}, S)$, only rule will change.

Removing ϵ-Productions

- **How do we construct $G^{'}$?**
- First find out the **nullable non-terminals**.
  - A non-terminals A is said to be **nullable**, if it is possible to derive ϵ from that non-terminals ($A \overset{*}{\Rightarrow} \epsilon$).
  - If we consider a grammar $A \rightarrow BC$, $B \rightarrow \epsilon$, $C \rightarrow \epsilon$, then A, B, C all are nullable.
- If it is nullable non-terminal then you have two possibilities, either you can keep it or you can remove it.

Removing ϵ-Productions

- **Example :** Find a context-free grammar without ϵ-productions equivalent to the grammar defined by :

$$\begin{aligned}
S &\to ABaC, \\
A &\to BC, \\
B &\to b|\epsilon, \\
C &\to D|\epsilon, \\
D &\to d,
\end{aligned}$$

- We find that the nullable non-terminals are A, B, C. Then we get the following:

$$\begin{aligned}
S &\to ABaC|BaC|AaC|ABa|aC|Aa|Ba|a, \\
A &\to B|C|BC, \\
B &\to b, \\
C &\to D, \\
D &\to d
\end{aligned}$$

Removing Unit-Productions

- A **unit production** is of the form $A \to B$. A non-terminal going into another non-terminal. Such a production is called a unit production.
- We want to get rid of these rules, because if we introduced too many unit production, the compiler time will be more. So, we want to avoid too many of this unit production.

Removing Unit-Productions

- A **unit production** is of the form $A \to B$. A non-terminal going into another non-terminal. Such a production is called a unit production.
- We want to get rid of these rules, because if we introduced too many unit production, the compiler time will be more. So, we want to avoid too many of this unit production.
- How to remove these unit productions ?

Removing Unit-Productions

- Find all pairs (A, B) such that $A \overset{*}{\Rightarrow} B$.
- Staring with the grammar $G = (N, T, P, S)$ and we want to remove the unit production without affecting the language generated.
- Now, split P into two sets, i.e., $P = P_1 \cup P_2$, where P_1 is the set of unit productions and P_2 is the set of non-unit productions.
- We can construct an equivalent grammar $G^{'} = (N, T, P^{'}, S)$, where only the production rules are different. So, what is $P^{'}$?
  - $P^{'}$ consists of all P_2, all P_1 will be removed but we add some more rules. What are the rules we going to add ?
  - For every pair (A, B) such that $A \overset{*}{\Rightarrow} B$, add rules $A \to \beta$ if $B \to \beta \in P_2$.
- Then we can show that $L(G) = L(G^{'})$.
  - That means, by removing the unit productions and adding the rules of the form $A \to \beta$, the resultant language is not changed.

Removing Unit-Productions

- **Example :** Remove all unit-productions from the following grammar–

$$\begin{aligned} S &\rightarrow Aa|B, \\ B &\rightarrow A|bb, \\ A &\rightarrow a|bc|B, \end{aligned}$$

Removing Unit-Productions

- **Example :** Remove all unit-productions from the following grammar–

$$\begin{aligned} S &\rightarrow Aa|B, \\ B &\rightarrow A|bb, \\ A &\rightarrow a|bc|B, \end{aligned}$$

- Here, the unit productions are $S \rightarrow B$, $B \rightarrow A$, $A \rightarrow B$ and non-unit productions are $S \rightarrow Aa$, $B \rightarrow bb$, $A \rightarrow a|bc$

Removing Unit-Productions

- **Example :** Remove all unit-productions from the following grammar–

$$\begin{aligned} S &\rightarrow Aa|B, \\ B &\rightarrow A|bb, \\ A &\rightarrow a|bc|B, \end{aligned}$$

- Here, the unit productions are $S \rightarrow B$, $B \rightarrow A$, $A \rightarrow B$ and non-unit productions are $S \rightarrow Aa$, $B \rightarrow bb$, $A \rightarrow a|bc$
- So, $S \overset{*}{\Rightarrow} A$, $S \overset{*}{\Rightarrow} B$, $B \overset{*}{\Rightarrow} A$, $A \overset{*}{\Rightarrow} B$

Removing Unit-Productions

- **Example :** Remove all unit-productions from the following grammar–

$$\begin{aligned} S &\rightarrow Aa|B, \\ B &\rightarrow A|bb, \\ A &\rightarrow a|bc|B, \end{aligned}$$

- Here, the unit productions are $S \rightarrow B$, $B \rightarrow A$, $A \rightarrow B$ and non-unit productions are $S \rightarrow Aa$, $B \rightarrow bb$, $A \rightarrow a|bc$
- So, $S \overset{*}{\Rightarrow} A$, $S \overset{*}{\Rightarrow} B$, $B \overset{*}{\Rightarrow} A$, $A \overset{*}{\Rightarrow} B$
- The new rules are–

$$\begin{aligned} S &\rightarrow a|bc|bb, \\ A &\rightarrow bb, \\ B &\rightarrow a|bc \end{aligned}$$

Removing Unit-Productions

- **Example :** Remove all unit-productions from the following grammar–

$$\begin{aligned} S &\rightarrow Aa|B, \\ B &\rightarrow A|bb, \\ A &\rightarrow a|bc|B, \end{aligned}$$

- Here, the unit productions are $S \rightarrow B$, $B \rightarrow A$, $A \rightarrow B$ and non-unit productions are $S \rightarrow Aa$, $B \rightarrow bb$, $A \rightarrow a|bc$
- So, $S \overset{*}{\Rightarrow} A$, $S \overset{*}{\Rightarrow} B$, $B \overset{*}{\Rightarrow} A$, $A \overset{*}{\Rightarrow} B$
- The new rules are–

$$\begin{aligned} S &\rightarrow a|bc|bb, \\ A &\rightarrow bb, \\ B &\rightarrow a|bc \end{aligned}$$

- The equivalent grammar is–

$$\begin{aligned} S &\rightarrow a|bc|bb|Aa, \\ A &\rightarrow a|bc|bb, \\ B &\rightarrow a|bc|bb \end{aligned}$$

Reduced Grammar

- **Theorem :** Let L be a context-free language that does not contain ϵ. Then there exixts a context-free grammar that generates L and that does not have any uselss productions, ϵ-productions, or unit-productions.
- In which order we go ?
  - First step will be remove the ϵ-productions, because when we remove the ϵ-productions we may introduced unit-productions.
  - After removing ϵ-productions, remove the unit-productions.
  - Then remove the useless productions using Lemma 1 and Lemma 2 in that order.
  - Ultimately, we will end up with a grammar which does not have ϵ-productions, unit-productions and useless productions. Such a grammar is called **reduced grammar**.

Reduced Grammar

- $L - \{\epsilon\}$ is generated by a reduced grammar. Now, we have to include ϵ, for that what we do ?
- Suppose $G = (N, T, P, S)$, then we have $G^{'} = (N \cup \{S_1\}, T, P \cup P_1, S_1)$, we add one more symbol and make it start symbol.
- All the productions will be there, but we may add a few more productions. Now, what is the set of productions P_1 we are going to add ?
  - We want to inlude ϵ in the language. So, we have a rule $S_1 \rightarrow \epsilon$. So, if we want to drive ϵ, we will use $S_1 \rightarrow \epsilon$ rule alone.
  - If $S \rightarrow \alpha \in P$, add $S_1 \rightarrow \alpha$ to P_1. This make sure that the start symbol does not occur on the right hand side.