**Unit Productions**

**Example 1:**

S → 0A | 1B | C

A → 0S | 00

B → 1 | A

C → 01

S → C is a unit production. But while removing S → C we have to consider what C gives. So, we can add a rule to S.

S → 0A | 1B | 01

Similarly, B → A is also a unit production so we can modify it as

B → 1 | 0S | 00

Thus finally we can write CFG without unit production as

S → 0A | 1B | 01

A → 0S | 00

B → 1 | 0S | 00

C → 01

**Example 2:**

S -> AB  
A -> a  
B -> C / b  
C -> D  
D -> E  
E -> a  
**Solution:**  
There are 3 unit production in the grammar  
B -> C  
C -> D  
D -> E  
For production D -> E there is E -> a so we add D -> a to the grammar and add D -> E from the grammar. Now we have C -> D so we add a production C -> a tp the grammar and delete C -> D from the grammar. Similarly we have B -> C by adding B -> a and removing B -> C we get the final grammar free of unit production as:  
S -> AB  
A -> a  
B -> a / b  
C -> a  
D -> a  
E -> a  
We can see that C, D and E are unreachable symbols so to get a completely reduced grammar we remove them from the CFG. The final CFG is :  
S -> AB  
A -> a  
B -> a / b

**Example 3:**

S -> S + T/ T  
T -> T \* F/ F  
F -> (S)/a

S -> T and T -> F are the two unit productions in the CFG.  
For productions T -> F we have F -> (S)/a so we add T -> (S)/a to the grammar and remove T-> F from the grammar. Now for production S -> T we have production T -> T \* F/(S)/a so we add S -> T \* F/(S)/a to the grammar. So the grammar after removal of unit production is:  
S->S + T/ T \* F/ (S)/ a  
T -> T \* F/ F  
F -> (S)/ a

**Example 4:**

Remove unit productions from a grammar (G1) whose production rule is given by

P: S→XY, X→a, Y →Z | b, Z→M, M→N, N→a                        // Grammar (G1)

In above grammar (G1) Unit Productions are

Y → | Z

Z→ M

M → N

The production unit which is removed easily is considered first. Let see,

**For the Removal of Third Unit Production (M →  N)**

As N→a So, Unit Production M → N is updated to M→a.

**For the Removal of Second Unit Production (Z → M)**

As we derived M→a in above case, So, Unit Production Z → M is updated to Z→a

**For the Removal of First Unit Production (Y →  Z)**

As we derived Z→a, So, Unit Production Y→Z is updated to Y→a

After Removal Unit Productions the **Updated Grammar (G2)** is given below

**P:** S→XY, X→a, Y →a| b, Z→a, M→a, N→a            // Grammar (G2)

We can**remove the unreachable states** from above grammar (G2). So Finally, Grammar (G2) is given below

**P:** S→XY, X→a, Y →a| b.                                         // Grammar (G2)

**Example 5:**

Remove unit productions from a grammar (G1) whose production rule is given by

P: S → aA | B, A → ba | bb, B → A | bba                       // Grammar (G1)

In above grammar Unit Production is

S → B

B→ A

The production unit which is removed easily is considered first. Let see,

For the Removal of 2nd Unit Production (B→ A)

As A→ba | bb. So, Unit Production B → A | bba is updated to B→ba | bb.

For the Removal of first Unit Production (S →  B)

As B→A | ba | bb and A →ba|bb Therefore B→ ba | bb | bba. So,  Unit Production S → B is updated to S→ ba | bb | bba.

After Removal Unit Productions the Updated Grammar (G2) is given below

P: S → aA | ba | bb | bba, A → ba | bb, B → A | bba                       // Grammar (G1)

We can remove the unreachable states from above grammar (G2). So Finally, Grammar (G2) is given below

P: S → aA | ba | bb | bba, A → ba | bb                       // Grammar (G2)

**Example 6:**

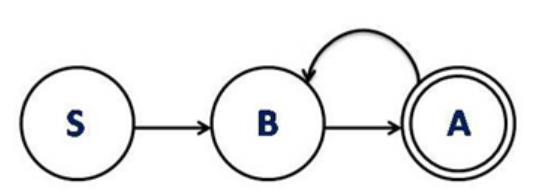
consider a grammar as an example.

G₁: S→ Aa|B, B→ A|bb, A → a|bc|B

Step 1:

First, we create a dependency graph of all unit production.

S→B, BA and A→B



So, SB,SA,B⇒ A and A ⇒ B

Step 2:

Now we write grammar without unit production:

S→ Aa S→ bb + S→ a|bc [Reason S⇒ B, S⇒ A ]

B→ bb + B → a|bc

[Reason B⇒ A]

A→ abc + A → bb

[Reason A⇒ B]

Whatever we derive from B, we same way derive from A because A ⇒ B, and same things happen for all production.

New grammar:

G2: S→ Aa|bb|a|bc

A→ a|bc|bb

B→ bb|a|bc

So, G₁ = G₂ and L(G₁) = L(G2)

**Example 7:**

S -> Aa | B

A -> b | B

B -> A | a

Lets add all the non-unit productions of ‘G’ in ‘Guf’. ‘Guf’ now becomes – 

S -> Aa

A -> b

B -> a

Now we find all the variables that satisfy ‘X \*=> Z’. These are ‘S\*=>B’, ‘A \*=> B’ and ‘B \*=> A’. For ‘A \*=> B’ ,

we add ‘A -> a’ because ‘B ->a’ exists in ‘Guf’. ‘Guf’ now becomes 

S -> Aa

A -> b | a

B -> a

For ‘B \*=> A’ , we add ‘B -> b’ because ‘A -> b’ exists in ‘Guf’. The new grammar now becomes 

S -> Aa

A -> b | a

B -> a | b

We follow the same step for ‘S\*=>B’ and finally get the following grammar – 

S -> Aa | b | a

A -> b | a

B -> a | b

Now remove B -> a|b , since it doesnt occur in the production ‘S’, then the following grammar becomes,

S->Aa|b|a

A->b|a