Week 8: Lecture Notes

Context free Grammars (CFG)

A context free grammar is $G = (V, T, P, S)$ where

- V is a finite set (non-empty) whose elements are called variables
- T is a finite set (non-empty) whose elements are called terminals
- $V \cap T = \phi$
- S is a special variable called the start variable
- P is a finite set of productions or rules whose elements are $A \to \alpha$ where A is a variable and α is a string of symbols from $(V \cup T)^*$

Example:

$G = (\{E\}, \{+, *, (,), id\}, P, E)$

where P represents the following set of rules

$E \to E+E$ $\quad$ $E \to (E)$

$E \to E*E$ $\quad$ $E \to id$.

Applying productions repeatedly, we can obtain more and more complicated expressions.

$\Rightarrow$ denotes act of deriving

$(id+id)*id$ consists of only terminal symbols, it is a word in the language generated by G.

$E \Rightarrow E+E$
$\Rightarrow (E)*E$
$\Rightarrow (E)*id$
$\Rightarrow (E+E)*id$
$\Rightarrow (E+id)*id$
$\Rightarrow (id+id)*id$

Example:

L_{pal} : languages of palindromes of 0's and 1's
- not regular

- $\varepsilon, 0, 1$ are palindromes
- if w is a palindrome, so are $0w0$ and $1w1$

Such recursively defined languages (L_{pal}) can be expressed formally as context free grammar (for palindromes) G_{pal}

$$G_{pal} = (\{E\}, \{0,1\}, P, E)$$

where P represents the set of five productions

$E \to E$ $\quad$ $E \to 0E0$

$E \to 1$ $\quad$ $E \to 1E1$

$E \to 0$

- The set of productions is the kernel of grammar and language specifications

- Note (Regarding production rules)

(i) Reverse substition is not permitted

e.g. if $S \to AB$ is a production, then we can replace S by AB but <u>we cannot replace</u> AB by S

(ii) No inversion operation is permitted

eg. if $S \to AB$ is a production, then it is not necessary that $AB \to S$ is a production.

Notations

1. The capita letters A, B, C, D, E and S denote variables. S is the start symbol
2. The lowercase letters a, b, c, d, e, digits, boldface strings are terminals.
3. The capital letters X, Y, Z denote symbols that may be either terminals or variables.
4. The lowercase letters u, v, w, x, y, z denote strings of terminals.
5. The lower case Greek letters α, β, γ denote strings of variables and terminals $(V \cup T)^*$

A CFG is often presented by simply listing its productions.

If $A \to \alpha_1, A \to \alpha_2, \ldots, A \to \alpha_k$ are productions for variable , then we may express them by the notation

$A \to \alpha_1 | \alpha_2 | \ldots | \alpha_k$ '|' stands for 'or'

Example:

$E \to E+E$, $E \to E*E$, $E \to (E)$, $E \to id$

can be written as

$E \to E+E \mid E*E \mid (E) \mid id$

Derivations and languages

- $G = (V, T, P, S)$ → CFG
- $\underset{G}{\Rightarrow}$ → relation on $(VUT)^*$

defined as follows:

if $A \to \beta$ is a production of P and $\alpha, \gamma \in (VUT)^*$

then $\alpha A \gamma \underset{G}{\Rightarrow} \alpha \beta \gamma$

$\alpha A \gamma$ directly derives $\alpha \beta \gamma$ in grammar G.

- Two strings are related by $\underset{G}{\Rightarrow}$ exactly when the second is obtained from the first by one application of some production

- $\overset{*}{\underset{G}{\Rightarrow}}$ relation on $(VUT)^*$

defined as follows:

Suppose that $\alpha_1, \alpha_2, \ldots \alpha_m, \in (VUT)^*$, $m \geq 1$ and

$\alpha_1 \underset{G}{\Rightarrow} \alpha_2$, $\alpha_2 \underset{G}{\Rightarrow} \alpha_3$, $\ldots$ $\alpha_{m-1} \underset{G}{\Rightarrow} \alpha_m$

Then $\alpha_1 \overset{*}{\underset{G}{\Rightarrow}} \alpha_m$ or α_1 derives α_m in grammar G

- $\overset{*}{\underset{G}{\Rightarrow}}$ is reflexive, transitive closure of $\underset{G}{\Rightarrow}$
- $\alpha \overset{*}{\underset{G}{\Rightarrow}} \alpha$ for each string $\alpha \in (VUT)^*$

- When it is clear which grammar G is involved, we write $\Rightarrow$ instead of $\underset{G}{\Rightarrow}$ and $\overset{*}{\Rightarrow}$ instead of $\overset{*}{\underset{G}{\Rightarrow}}$
- $\alpha \overset{i}{\Rightarrow} \beta$ if α derives β by exactly i steps

- The language generated by G is

$$L(G) = \{w \mid w \in T^* \text{ and } S \overset{*}{\underset{G}{\Rightarrow}} w\}$$

i.e. a string is in $L(G)$ if

i. The string consists of solely of terminals
ii. The string can be derived from S

Context free language (CFL)

L is a context free language (CFL) if $L = L(G)$ for some CFG G

Sentential form

A string $\alpha \in (VUT)^*$ is called a sentential form if $S \overset{*}{\Rightarrow} \alpha$

Equivalent grammar

G_1, G_2 are equivalent if $L(G_1) = L(G_2)$

Example:

Consider CFG $G = (V, T, P, S)$ where

$V = \{S\}$, $T = \{a, b\}$, $P = \{S \to aSb, S \to ab\}$. Find $L(G)$

$$L(G) = \{w \mid w \in T^* \text{ and } S \overset{*}{\underset{G}{\Rightarrow}} w\}$$

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow a^3Sb^3 \Rightarrow \dots \Rightarrow a^{n-1}Sb^{n-1}$$

(applying $S \to aSb$ $n-1$ times)

$$\Rightarrow a^n b^n \quad \text{(applying } S \to ab\text{)}$$

Thus, $S \overset{*}{\Rightarrow} a^n b^n \in T^* \Rightarrow a^n b^n \in L(G)$ for $n \geq 1$

Now, we need to show that $L(G) \subseteq \{a^n b^n \mid n \geq 1\}$

~~Let $\alpha \in L(G)$~~.

Let $w \in L(G)$. w is a terminal string i.e. $w \in T^*$

The only way w can be derived from start symbol S is

$$S \overset{*}{\Rightarrow} a^{n-1}Sb^{n-1} \Rightarrow a^n b^n \quad \text{for some } n \geq 1$$

- $S \overset{*}{\Rightarrow} a^{n-1}Sb^{n-1}$: applying $S \to aSb$ few no. of times
  - ⇓ number of S's remains same in the sentential form
  - ⇓ i.e. $L(G) \subseteq \{a^n b^n \mid n \geq 1\}$
- $\Rightarrow a^n b^n$: one application of $S \to ab$
  - ⇓ number of S's is reduced by 1 in the sentential form
  - ⇓ no S remains in the resulting string

Thus

$$L(G) = \{a^n b^n \mid n \geq 1\}$$

Example

Find a CFG generating the language of even length palindromes.

i.e. $L = \{ww^R \mid w \in \Sigma^* = \{a,b\}^*\}$

$G = (\{S\}, \{a,b\}, P, S)$,

P is defined as $S \to aSa \mid bSb \mid \varepsilon$

Arguments:

i. ε is a palindrome

ii. a, b are palindromes

iii. if x is a palindromes, then axa, bxb are palindromes.

Example:

Construct a CFG generating

$$L = \{wcw^T : w \in \{a,b\}^*\}$$

$G = (\{S\}, \{a,b,c\}, P, S)$

where P is defined as $S \to c \mid aSa \mid bSb$

Argument:

Any string in L is generated by recursion as follows:

i. $c \in L$

ii. if $x \in L$, then $wxw^T \in L$

Example

A CFG for the regular language corresponding to the regular expression 00^*11^*,

$$L = \{ 0^i 1^j \mid i, j > 0 \}$$

idea:

The language is the concatenation of two languages

- language of strings of 0's
- language of strings of 1's

$G = (\{S, C, D\}, \{0, 1\}, P, S)$

where P is defined as $\{ S \to CD,\ C \to 0C \mid 0,\ D \to 1D \mid 1 \}$

Example:

CFG that generates sentences as composed of noun and verb phrases.

$S \to NP\ VP$

$NP \to \text{the}\ N$

$VP \to V\ NP$

$V \to \text{sings} \mid \text{eats}$

$N \to \text{cat} \mid \text{song} \mid \text{bird}$

"the bird sings the song" → accepted

"the song eats the cat" → also accepted

- All legal sentences are generated, not just those that are meaningful — content-free

Example:

Complement of $\{0^i 1^j : i,j > 0\}$

idea:

partition the set of strings into three failure cases

1. strings not of the right form : somewhere there is a 1 followed by a 0
2. only zeroes
3. only ones

$G = (\{S, A, B, C, D\}, \{0,1\}, P, S)$

where P has the following production rules:

$S \to A|B|C$

$A \to D01D$ (produces all strings with a 0 and 1 out of order)

$D \to 0D|1D|\varepsilon$ (produces all strings)

$B \to 0B|0$ (produces strings of 0's)

$C \to 1C|1$ (produces strings of 1's)

e.g.

Test whether $w \in L(G)$ where

$w = 101001$

$S \Rightarrow A \Rightarrow D10D \Rightarrow \varepsilon 10D = 10D$

$\Rightarrow 101D$

$\Rightarrow 1010D$

$\Rightarrow 10100D$

$\Rightarrow 101001D$

$\Rightarrow 101001\varepsilon$

$\Rightarrow 101001$

$= w$

$\therefore w \in L(G) = \{w | w \in T^* \text{ and } S \overset{*}{\Rightarrow} w\}$

Example:

Find a CFG G for all binary strings with an even number of 0's

$S \to 1S \mid 0A0S \mid \varepsilon$

$A \to 1A \mid \varepsilon$

($1A$ produces strings of 1's.)

How to decompose such strings?

1. 1st symbol starts with 1, followed by even no. of 0's
2. 1st symbol starts with 0, then go to the next 0, followed even number of 0's

e.g. $101000 \in L(G)$ as

$S \Rightarrow 1S \Rightarrow 10A0S \Rightarrow 101A0S \Rightarrow 101\varepsilon 0S = 1010S$

$\Rightarrow 10100A0 \Rightarrow 10100\varepsilon 0 \Rightarrow 101000$

- A language can be more than one grammar

$S \to S \mid 0T \mid \varepsilon$, $T \to T \mid 0S$ } also generates all binary strings with even no. of 0's

idea:

We decompose all binary strings with an even number of 0's as follows:

1. 1st symbol is 0, followed by odd number of
2. We use T to generate all binary strings with odd number of zeroes.
3. We use S to generate all binary string with even number of zeroes.

Production rules: $S \to 1S \mid 0T \mid \varepsilon$

$T \to 1T \mid 0S$

Example:

Let $G = (\{S, C\}, \{a, b\}, P, S)$ where P consists of $S \to aCa$, $C \to aCa \mid b$. Find $L(G)$.

$S \Rightarrow aCa \Rightarrow aba \quad \therefore aba \in L(G)$

$S \Rightarrow aCa \Rightarrow aaCaa \Rightarrow \dots \Rightarrow a^n C a^n \Rightarrow a^n b a^n$

$\therefore a^n b a^n \in L(G), \; n \geq 1$

$\therefore \{a^n b a^n \mid n \geq 1\} \subseteq L(G)$ ——— (1)

Now, to prove $L(G) \subseteq \{a^n b^* a^n \mid n \geq 1\}$

Any string $w \in T^* = \{a, b\}^*$ will be in $L(G)$ if

$$S \overset{*}{\Rightarrow} w$$

The only S-production is $S \to aCa$

if we apply $C \to b$, we get $aba \in T^*$

$$S \overset{*}{\Rightarrow} aba$$

if we apply $C \to aCa$, once or several times

we get $S \overset{*}{\Rightarrow} a^n C a^n$

$\Rightarrow a^n b a^n \in T^*$

(one application of $C \to b$)

Thus any derivation is of the form

$$S \overset{*}{\Rightarrow} a^n b a^n, \quad n \geq 1$$

$\Rightarrow L(G) \subseteq \{a^n b a^n \mid n \geq 1\}$ ——— (2)

Thus from (1) and (2)

$$L(G) = \{a^n b a^n \mid n \geq 1\}$$

Example:

Let $G = (\{S, A_1\}, \{0,1,2\}, P, S)$ where P consists of

$S \to 0SA_1 2$, $S \to 012$, $2A_1 \to A_1 2$, $1A_1 \to 11$. Find $L(G)$

We have

$S \Rightarrow 012 \quad \therefore \quad 012 \in L(G)$

$S \Rightarrow 0\underline{S}A_1 2 \Rightarrow 0012\underline{A_1}2 \Rightarrow 001\underline{A_1}22 \Rightarrow 001122$

$S \Rightarrow 0SA_1 2 \Rightarrow 00SA_1 2A_1 2 \Rightarrow 000SA_1 2A_1 2A_1 2$

$S \to 0SA_1 2$ applied $(n-1)$ times

$\overset{*}{\Rightarrow} 0^{n-1} S (A_1 2)^{n-1}$

$\Rightarrow 0^{n-1} 012 (A_1 2)^{n-1}$

$\Rightarrow 0^n 1\ 2A_1 (A_1)^{n-2} 2^{n-1}$

$\Rightarrow 0^n 1\ A_1 2\ A_1 (A_1)^{n-3} 2^{n-1}$

$\Rightarrow 0^n 1\ A_1 A_1\ 2A_1 (A_1)^{n-4} 2^{n-1}$

$\vdots$

$\Rightarrow 0^n 1\ A_1^{n-1} 2^n$

$\Rightarrow 0^n 1A_1\ A_1^{n-2} 2^n$

$\Rightarrow 0^n 11\ A_1^{n-2} 2^n$

$\Rightarrow 0^n 111\ A_1^{n-3} 2^n$

$\vdots$

$\Rightarrow 0^n 1^n 2^n$

$\therefore \quad 0^n 1^n 2^n \in L(G)$

$\Rightarrow \quad \{0^n 1^n 2^n \mid n \geq 1\} \subseteq L(G)$

Then we need to show that $L(G) \subseteq \{0^n 1^n 2^n \mid n \geq 1\}$

Example

Let $G = (\{S, A_1, A_2\}, \{a, b\}, P, S)$

where P consists of

$S \to a A_1 A_2 a$, $A_1 \to ba A_1 A_2 b$

$A_2 \to A_1 ab$, $A_1 \to baa$

$bA_2 b \to abab$

Test whether $w = baabbabaaaabbaba$ is in $L(G)$.

$S \Rightarrow \underline{a A_1} A_2 a$

$\Rightarrow baa \underline{A_2} a$

$\Rightarrow baa \underline{A_1} aba$

$\Rightarrow baa\, ba \underline{A_1} A_2\, baba$

$\Rightarrow baabbaa \underline{A_2}\, baba$

$\Rightarrow baabba \underline{aA_1} abbaba$

$\Rightarrow baabbabaaaabbaba = w$

$\therefore w \in L(G)$

Example

If the grammar G is given by the productions $S \to aSa \mid bSb \mid aa \mid bb \mid \varepsilon$, show that

i) $L(G)$ has no strings of odd length

ii) any string in $L(G)$ is of length $2n$, $n \geq 0$

iii) the number of strings of length $2n$ is 2^n

On application of each production (except $S \to \varepsilon$), a variable is replaced by 2 terminals and atmost one variable

Thus each step in any derivation increases the number of of terminals by 2 except that involving $S \to \varepsilon$.

Which explains (i) and (ii)

Now any string w of length $2n$ is of the form

$$a_1 a_2 \ldots a_n a_n a_{n-1} \ldots a_1$$

where each a_i is either a or b

$\Rightarrow$ number of $2n$-bit strings in $L(G)$ is 2^n

Example

Construct a context free grammar G generating all integers (with sign)

Let $G = (V, T, P, S)$ where

$V = \{ S, \langle sign \rangle, \langle digit \rangle, \langle integer \rangle \}$

$T = \{ 0, 1, \ldots, 9, +, - \}$

and P consists of

$S \rightarrow \langle sign \rangle \langle integer \rangle$

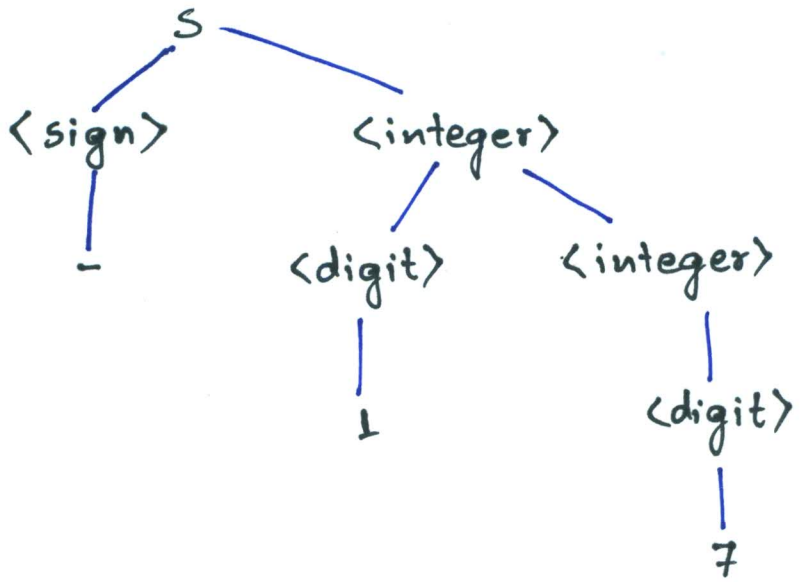
$\langle sign \rangle \rightarrow + \mid -$

$\langle integer \rangle \rightarrow \langle digit \rangle \langle integer \rangle \mid \langle digit \rangle$

$\langle digit \rangle \rightarrow 0 \mid 1 \mid 2 \mid \cdots \mid 9$

Then $L(G)$: the set of all integers.

e.g. derivation of -17



Derivation Trees / Parse Trees

- $G = (V, T, P, S)$ be a CFG. A tree is a **derivation tree or parse tree** for G if:

  i. Every vertex has a label, which is a symbol of
  $$V \cup T \cup \{\varepsilon\}$$

  ii. The label of the root is S

  iii. If a vertex is interior and has label A, then A must be in V

  iv. If vertex n has label A and vertices $n_1, n_2, \ldots, n_k$ are sons of vertex n, in order from the left with labels $X_1, X_2, \ldots, X_k$ respectively, then
  $$A \to X_1 X_2 \ldots X_k$$
  must be a production in P

  v. If vertex n has label ε, then n is a leaf and is the only son of its father

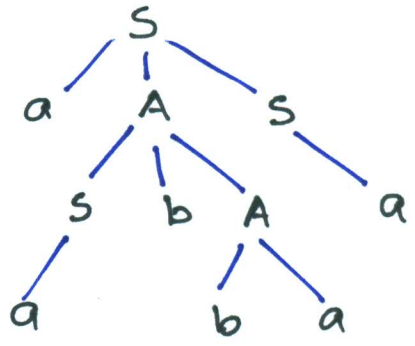
Example:

$G = (\{S, A\}, \{a, b\}, P, S)$

where consists of

$S \to aAS \mid a$

$A \to SbA \mid SS \mid ba$



A natural description of the sentential form

yield of the derivation tree $\longrightarrow$ aabbaa

- The yield of a derivation tree is the concatenation of the labels of the leaves without repetition in the left-to-right ordering.

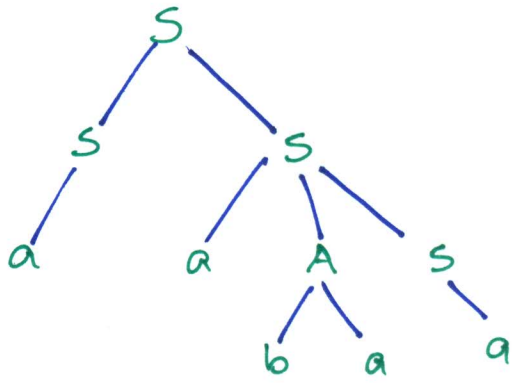
Example:

$G = (\{S, A\}, \{a, b\}, P, S)$

where P consists of

$S \rightarrow aAS \mid a \mid SS$

$A \rightarrow SbA \mid ba$



A parse tree showing the derivation

$S \overset{*}{\Rightarrow} aabaa$

- The yield of the derivation tree is a sentential form in G. i.e. yield $\in (V \cup T)^*$

Example:

$G = (\{E, I\}, T, P, E)$

where P consists of

$E \rightarrow I \mid E+E \mid E*E \mid (E)$

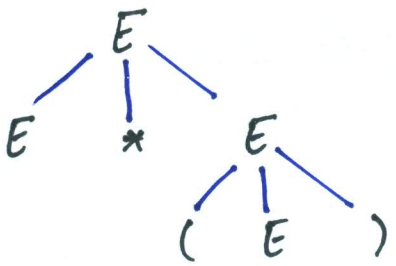
$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

and

$T = \{+, *, (,), a, b, 0, 1\}$

(i) A parse tree showing the derivation

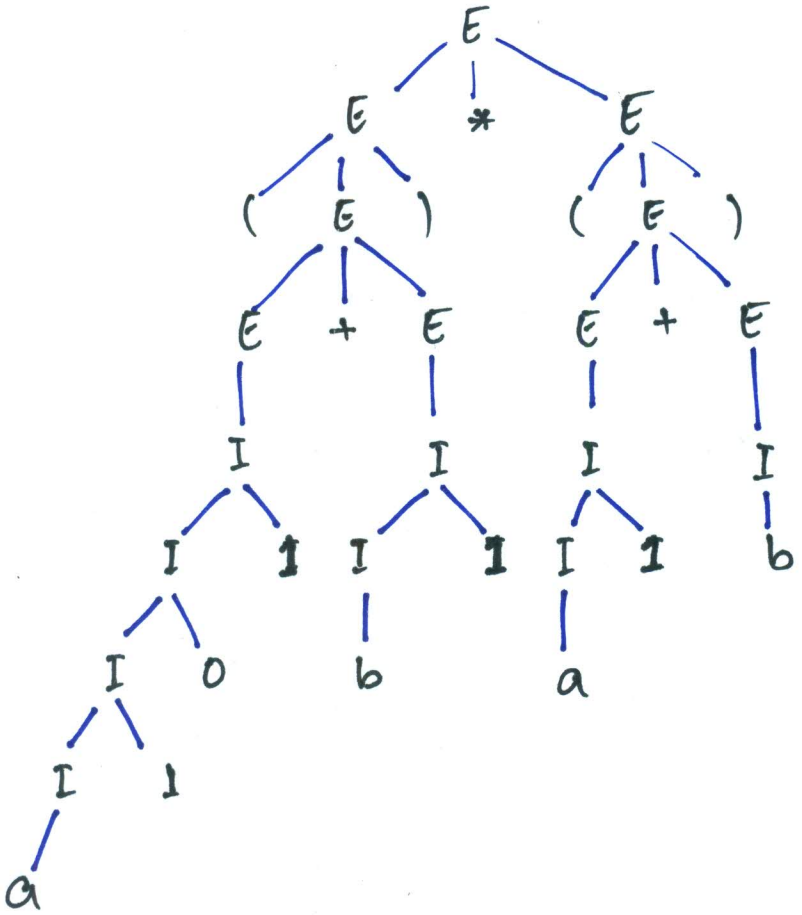
$E \overset{*}{\Rightarrow} E * (E)$



yield $\in (VUT)^*$

A parse tree showing the derivation

$E \overset{*}{\Rightarrow} (a101 + b1) * (a1 + b)$



yield $\in T^*$

Relationship between derivation trees and derivations

Theorem:

Let $G = (V, T, P, S)$ be a context-free grammar. Then $S \overset{*}{\Rightarrow} \alpha$ iff there is a derivation tree in grammar G with yield α

Proof:

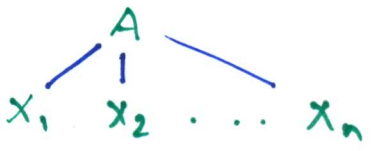
A-tree: A subtree of a derivation tree is called an A-tree if the label of its root is A

S-tree: derivation tree when S is the start symbol

We prove that $A \overset{*}{\Rightarrow} \alpha$ iff there is an A-tree with yield α.

(if) Suppose α is the yield of an A-tree. We prove by induction on K = the no. of internal vertices in the tree, that $A \overset{*}{\Rightarrow} \alpha$

Basis: $K=1$, i.e. only one internal vertex



$\therefore \alpha = X_1 X_2 \ldots X_n$ and

$A \to \alpha$ must be a production rule in G by definition of a derivative tree

Induction step:

Let us assume the result for all derivation trees with $\leq K-1$ internal vertices for some $K > 1$

Let α be the yield of an A-tree with K- interior vertices for some $K > 1$

Let the sons of the root A have labels $X_1, X_2, \ldots, X_n$ in order from the left

Then $A \to X_1 X_2 \ldots X_n$ is in P ($n \geq 1$ is any integer)

- $X_1, X_2, \ldots X_n$ cannot all be leaves (as $K > 1$)
- if i-th son of A is not a leaf, then it is the root of a subtree and X_i must be a variable.

X_i - tree with yield α_i say.

if i^{th} son of A is a leaf, let $\alpha_i = X_i$

Then $\alpha = \alpha_1 \alpha_2 \ldots \alpha_n$ as one can easily see that if $j < i$, then vertex j and all its descendants are to the left of vertex i and all its descendants.

Now subtrees must have fewer interior nodes than its tree does, unless the subtree is the entire tree.

$\therefore$ We have for each X_i-tree with yield α_i

$X_i \overset{*}{\Rightarrow} \alpha_i$ by induction as X_i- tree is not the entire tree.

Also, $X_i \Rightarrow \alpha_i$ if $X_i = \alpha_i$

$\therefore A \Rightarrow X_1 X_2 \ldots X_n \Rightarrow \alpha_1 X_2 \cdots X_n$

$\overset{*}{\Rightarrow} \alpha_1 \alpha_2 \cdots X_n \Rightarrow \ldots$

$\overset{*}{\Rightarrow} \alpha_1 \alpha_2 \ldots \alpha_n = \alpha$

Thus $A \overset{*}{\Rightarrow} \alpha$

(Only if)

Let $A \overset{*}{\Rightarrow} \alpha$

We must show that there is an A-tree with yield α

We do it by induction on K = the number of steps in $A \overset{*}{\Rightarrow} \alpha$

(Base)

K = 1

if $A \overset{*}{\Rightarrow} \alpha$ in single step, then $A \to \alpha$ is in.

Induction:

Assume the result is true for derivatives in $< K$ steps

Let $A \overset{K}{\Rightarrow} \alpha$

This can be splitted as $\underbrace{A \Rightarrow X_1 X_2 \ldots X_n}_{\text{implies } A \to X_1 X_2 \ldots X_n \text{ is in P}} \overset{K-1}{\Rightarrow} \alpha$

- In $X_1 X_2 \ldots X_n \overset{K-1}{\Rightarrow} \alpha$ either

  i. X_i is not changed throughout the derivation

  or ii. X_i is changed in some subsequent step.

Let α_i be the substring derived from X_i

Then $X_i = \alpha_i$ is in (i) and $X_i \overset{*}{\Rightarrow} \alpha_i$ is in (ii)

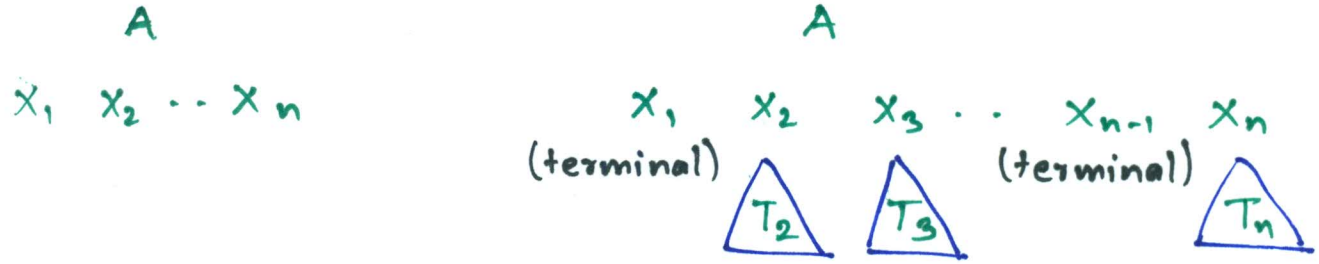
- As G is context free, in every step of derivation $X_1 X_2 \dots X_n \overset{*}{\Rightarrow} \alpha$, we replace a single variable by a string

  As $\alpha_1, \alpha_2, \dots \alpha_n$ accounts for all the symbols in α we have $\alpha = \alpha_1 \alpha_2 \dots \alpha_n$

- If X_i is a variable, then the derivation $X_i \overset{*}{\Rightarrow} \alpha_i$ has $< K$ steps as the entire derivation $A \overset{*}{\Rightarrow} \alpha$ takes K steps and the first step is surely not a part of $X_i \overset{*}{\Rightarrow} \alpha_i$

Hence by induction, for each X_i that is a variable, there is an X_i-tree T_i with yield α_i

Now, we construct an <u>A-tree</u> with <u>yield α</u> as follows:



First construct an A-tree with n leaves labeled $X_1, X_2, \dots X_n$ and no other vertex X

Replace each vertex with label X_i by tree T_i where X_i is non-terminal.

If X_i is a terminal vertex, no replacement is made.

The yield of the resulting A-tree is $\alpha_1 \alpha_2 \dots \alpha_n = \alpha$

Note:

The derivation tree does not specify the order in which we apply the productions for getting α.

So, the same derivation tree can induce several derivations (with the same yield)

Remark:

Let $A \overset{*}{\Rightarrow} w$ where $w \in T^*$ (a terminal string)

and $A \Rightarrow A_1 A_2 \cdots A_n$

Then we can write $w = w_1 w_2 \cdots w_n$

so that $A_i \overset{*}{\Rightarrow} w_i$

Week 9: Lecture Notes.

Theorem:

Let $G = (V, T, P, S)$ be a context free grammar. Then $S \overset{*}{\Rightarrow} \alpha$ iff there is a derivation tree in grammar G with yield α.

Example:

Consider CFG G whose productions are

$S \to aAS \mid a$, $A \to SbA \mid ss \mid ba$. Show that $S \overset{*}{\Rightarrow} aabbaa$ and construct a derivation tree whose yield is $aabbaa$

Soln:

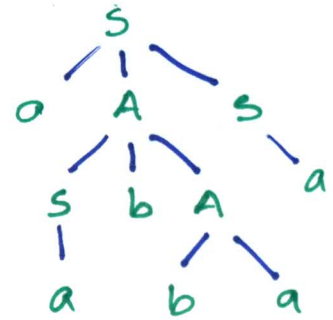
$S \Rightarrow aAS \Rightarrow aSbAS \Rightarrow aabAS \Rightarrow aabbaS \Rightarrow aabbaa$

another derivation:

$S \Rightarrow aAS \Rightarrow aAa$

$\Rightarrow aSbAa \Rightarrow aabAa \Rightarrow aabbaa$

$\Rightarrow aSbAa \Rightarrow aSbbaa \Rightarrow aabbaa$



Leftmost and rightmost derivations: ambiguity

- A derivation $A \overset{*}{\Rightarrow} w$ is called a leftmost derivation if we apply a production only to the leftmost variable at every step
- A derivation $A \overset{*}{\Rightarrow} w$ is a rightmost derivation if we apply a production only to the rightmost variable at every step.

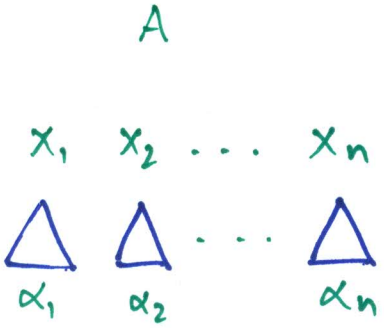
Theorem:

If $A \overset{*}{\Rightarrow} w$ in G, then there is a leftmost derivation of w

Proof:

By induction on the number of steps in $A \overset{*}{\Rightarrow} w$, we can prove it.

- If $w \in L(G)$ for CFG G, then w has atleast one parse tree and corresponding to a particular parse tree, w has a unique leftmost and a unique rightmost derivation



$A \Rightarrow X_1 X_2 \ldots X_n$
$\overset{*}{\Rightarrow} \alpha_1 X_2 \ldots X_n$
$\overset{*}{\Rightarrow} \alpha_1 \alpha_2 \ldots X_n$
$\vdots$
$\Rightarrow \alpha_1 \alpha_2 \ldots \alpha_n$

} derivation $X_i \overset{*}{\Rightarrow} \alpha_i$ are recursively made left most

$A \Rightarrow X_1 X_2 \ldots X_n$
$\overset{*}{\Rightarrow} X_1 X_2 \ldots X_{n-1} \alpha_n$
$\vdots$
$\Rightarrow \alpha_1 \alpha_2 \ldots \alpha_n$

} derivation $X_i \overset{*}{\Rightarrow} \alpha_i$ are recursively made right most

- w may have several leftmost or rightmost derivations as they may be more than one parsetree for w

Example:

Let G be the CFG. $S \to 0B \mid 1A$, $A \to 0 \mid 0S \mid 1AA$.
$B \to 1 \mid 1S \mid 0BB$.

for the string 00110101, find

a. the leftmost derivation

b. the rightmost derivation

c. the derivation tree

a). $S \Rightarrow 0\underline{B} \Rightarrow 00\underline{B}B \Rightarrow 001\underline{S}B \Rightarrow 0011\underline{A}B \Rightarrow 00110\underline{1}AB \Rightarrow$

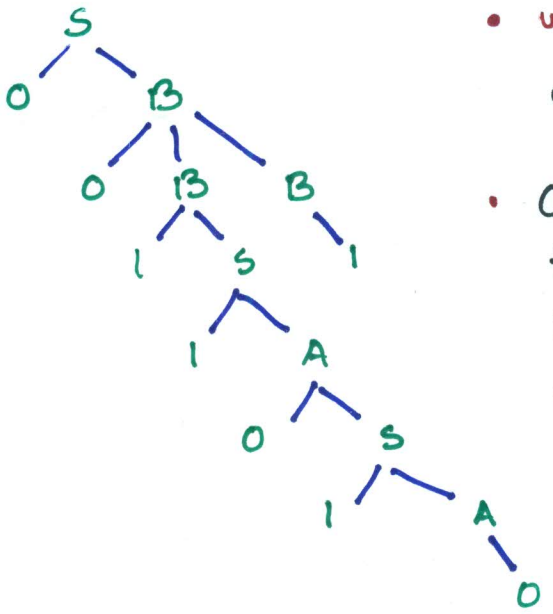
$\Rightarrow 00110\underline{1}0B \Rightarrow 00110101$

b). Corresponding rightmost derivation

$S \Rightarrow 0\underline{B} \Rightarrow 00B\underline{B} \Rightarrow 00\underline{B}1 \Rightarrow 001\underline{S}1 \Rightarrow 0011\underline{A}1 \Rightarrow$

$\Rightarrow 00110\underline{S}1 \Rightarrow 00110\underline{1}A1 \Rightarrow 00110101$

c). The derivation tree for w



- w may have several derivation trees
- Corresponding to each parse tree of w we have a unique leftmost derivation and a unique rightmost derivation

Another leftmost derivation:

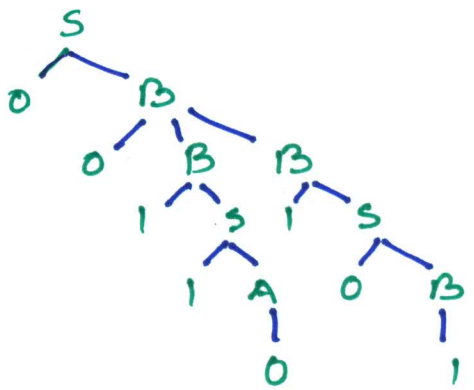
$S \Rightarrow 0B \Rightarrow 00BB \Rightarrow 001SB$

$\Downarrow$

$0011015 \Leftarrow 00110B \Leftarrow 0011AB$

$\Downarrow$

$00110101B \Rightarrow 00110101$



Corresponding rightmost derivation

$S \Rightarrow 0b \Rightarrow 00BB \Rightarrow 00B1S \Rightarrow 00B10B \Rightarrow 00B101$

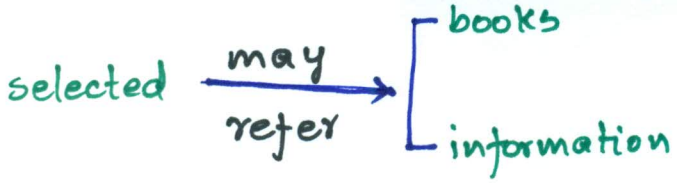
$\Downarrow$

$00110101 \Leftarrow 0011A101 \Leftarrow 001S101$

Ambiguity in CFG

Example:

"In books selected information is given"



↓ ambiguous sentence

i.e. sentence may be parsed in two different ways.

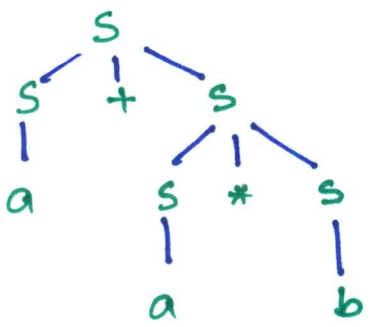
- A terminal string $w \in L(G)$ is ambiguous if $\exists$ two or more derivation trees for w.
- A CFG G is ambiguous if $\exists$ some $w \in L(G)$, which is ambiguous.
- A CFL for which every CFG is ambiguous is said to be an inherently ambiguous CFL.

Example:

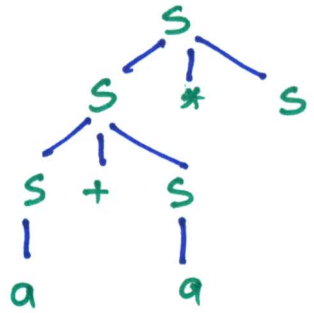
$G = (\{S\}, \{a, b, +, *\}, P, S)$, where P consists of

$$S \to S+S \mid S*S \mid a \mid b$$

Parse trees for $a + a * b$



T_1



T_2

The leftmost derivations of $a+a*b$ induced by T_1, T_2

- $S \Rightarrow S+S \Rightarrow a+S \Rightarrow a+S*S \Rightarrow a+a*S \Rightarrow a+a*b$
- $S \Rightarrow S*S \Rightarrow S+S*S \Rightarrow a+S*S \Rightarrow a+a*S \Rightarrow a+a*b$

Thus $a+a*b$ is ambiguous

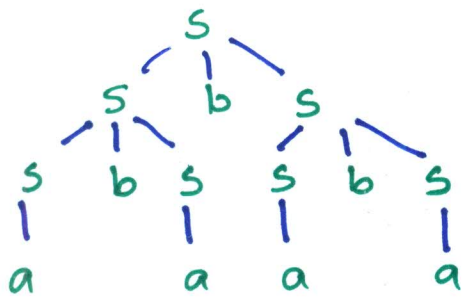
Example:

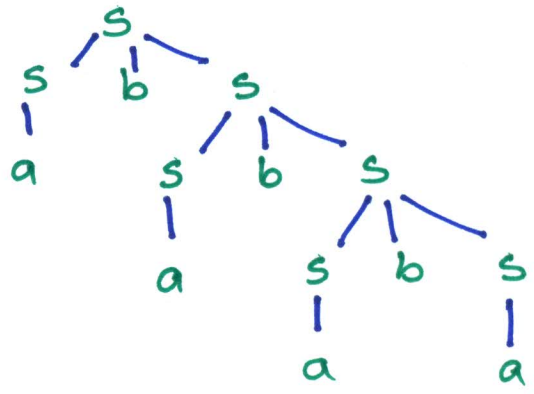
If G is the CFG $S \rightarrow SbS \mid a$. Show that G is ambiguous.

Soln:

We have to find a $w \in L(G)$ which is ambiguous.

Let $w = abababa$





Simplification of CFG's

- $G = (V, T, P, S)$ → CFG
- $L(G)$ generation → [may be not all symbols in VUT are used
  OR
  may be not all rules in P are used]
- Try to eliminate those symbols in VUT or rules in P from G and construct a CFG G' that realizes $L(G)$

Example:

$G = (\{S, A, B, D, E\}, \{a, b, c\}, P, S)$

where $P = \{S \to AB, A \to a, B \to b, B \to D, E \to c|\varepsilon\}$

Soln:

Note that $L(G) = \{ab\}$

Let $G' = (\{S, A, B\}, \{a, b\}, P', S)$

where $P' = \{S \to AB, A \to a, B \to b\}$

WHY?
- eliminate symbols D, E, c
- eliminate productions $B \to D, E \to c|\varepsilon$

1. D doesnot derive any terminal string
2. E and c doesnot appear in any sentential form
3. $E \to \varepsilon$ is a null production
4. $B \to D$ simply replaces B by D (unit production)

Algorithms to construct reduced grammars.

Construction 1

$G = (V, T, P, S)$

Define $G' = (V', T, P', S)$ as follows:

Construction of V'

We define $W_i \subseteq V$ by recursion:

$W_1 = \{ X \in V \mid$ there exists a production $X \to w$ where $w \in T^* \}$

[if $W_1 = \phi$; then some variable will remain after application of any production and so $L(G) = \phi$]

$W_{i+1} = W_i \cup \{ X \in V \mid$ there exists a production $A \to \alpha$ with $\alpha \in (T \cup W_i)^* \}$

Thus $W_i \subseteq W_{i+1} \ \forall \ i$

Also, $W_k = W_{k+1}$ for some $k \leq |V|$ as V is a finite set of variables.

Therefore $W_k = W_{k+j}$ for $j \geq 1$

We define $V' = W_k$

Construction of P'

$P' = \{ A \to \alpha \mid A, \alpha \in (V' \cup T)^* \}$

- $S \in V'$ (Prove it !!)

Example:

Let $G = (V, T, P, S)$ be given by the productions $S \to AB$, $A \to a$, $B \to b$, $B \to C$, $E \to d$. Find G' such that every variable in G' derives some terminal string.

Sol^n:

Construction of V'

$W_1 = \{A, B, E\}$

$W_2 = W_1 \cup \{X \in V \mid \exists$ some production $X \to \alpha$ with $\alpha \in (W_1 \cup T)^*\}$

$= W_1 \cup \{S\}$

$= \{A, B, E, S\}$

$W_3 = W_2 \cup \{X \in V \mid \exists$ some production $X \to \alpha$ with $\alpha \in (W_2 \cup T)^*\}$

$= W_2 \cup \phi$

$= W_2$

$\therefore V' = \{A, B, E, S\}$

Construction of P'

$P' = \{X \to \alpha \mid X, \alpha \in (V' \cup T)^*\}$

$= \{S \to AB, A \to a, B \to b, E \to d\}$

$\therefore G' = (\{S, A, B, E\}, \{a, b, d\}, P', S)$

Note:

$S \in V'$ always

Theorem 1

Given a CFG $G = (V, T, P, S)$, with $L(G) \neq \phi$, we can effectively find an equivalent CFG $G' = (V', T, P', S)$ such that for each A in V' there is some w in T^* for which $A \overset{*}{\Rightarrow} w$

Proof:

Let V', P' are constructed from G following the above algorithm.

Claim:

i) for each $A \in V'$, $A \overset{*}{\underset{G'}{\Rightarrow}} w$ for some $w \in T^*$

conversely $A \overset{*}{\underset{G'}{\Rightarrow}} w$ implies $A \in V'$

ii) $L(G) = L(G')$

Proof of claim (i)

Note that $W_K = W_1 \cup W_2 \cup \dots \cup W_K$

We prove by induction on i for $i = 1, 2, \dots K$ that $A \in W_i$ implies $A \overset{}{\underset{G'}{\Rightarrow}} w$ for some $w \in T^*$

Base:

Let $A \in W_1$

Then $A \to w$ is in P for some $w \in T^*$

By construction of P', $A \to w$ is in P'

This in turn implies $A \overset{*}{\underset{G'}{\Rightarrow}} w$

Induction:

Let the result be true for i

Consider $A \in W_{i+1}$

Then, either

a. $A \in W_i$, in which case $A \overset{*}{\underset{G}{\Rightarrow}} w$ for some $w \in T^*$ by induction

b. $\exists$ a production $A \to \alpha$ in P with $\alpha \in (T \cup W_i)^*$ in which case $A \to \alpha$ is in P' by construction of P'

Then α may be written as

$\alpha = X_1 X_2 \ldots X_n$ where $X_j \in T \cup W_i$

If $X_j \in W_i$, then by induction hypothesis,

$X_j \overset{*}{\underset{G'}{\Rightarrow}} w_j$ for some $w_j \in T^*$

Hence, $A \overset{*}{\underset{G'}{\Rightarrow}} w_1 w_2 \ldots w_n \in T^*$

Conversely we can show that $A \overset{*}{\underset{G'}{\Rightarrow}} w$ implies $A \in V'$

We can prove it by induction on K = number of steps in the derivation $A \overset{*}{\underset{G'}{\Rightarrow}} w$ in a similar manner.

Base: $K = 1$

production $A \to w$ is in P, so $A \in V'$

Induction

Let $A \underset{G}{\Rightarrow} X_1 X_2 \ldots X_n \overset{K-1}{\underset{G}{\Rightarrow}} w$ by a derivation of K steps.

We may write $w = w_1 w_2 \ldots w_n$ where $X_i \overset{*}{\Rightarrow} w_i$ for $1 \le i \le n$, by a derivation of fewer than K steps.

$\therefore$ By induction hypothesis $X_i \in V'$, $1 \le i \le n$

Hence, $A \to X_1 X_2 \ldots X_n$ is in P' by construction of P'.

Let $V' = W_1 \cup W_2 \cup \ldots \cup W_\ell$ and $W_\ell = W_{\ell+1}$

Let j be the least integer, $1 \le j \le \ell$ for which W_j contains all $X_1, X_2, \ldots, X_n$

As $A \to X_1 X_2 \ldots X_n$ is in P, where $X_1, X_2 \ldots X_n \in (T \cup W_j)^*$ we must have $A \in W_{j+1} \subseteq V'$

Thus $A \overset{K}{\underset{a}{\Rightarrow}} w$ implies $A \in V'$

Proof of claim (ii)

- $L(G') \subseteq L(G)$ as $P' \subseteq P$ and $V' \subseteq V$ ——— (1)
- To prove $L(G) \subseteq L(G')$, we need to prove the following auxilliary result.

$$A \overset{*}{\underset{G'}{\Rightarrow}} w \quad \text{if} \quad A \overset{*}{\underset{G}{\Rightarrow}} w \quad \text{for some } w \in T^*$$

We prove this by induction on K = the number of steps in the derivation $A \overset{*}{\underset{G}{\Rightarrow}} w$

Base:

$K=1$

Let $A \underset{G}{\Rightarrow} w$ where $w \in T^*$

Then $A \to w$ is in P'

$\therefore A \underset{G'}{\Rightarrow} w$

Induction

Result is true for $\leq K$ steps.

Let $A \overset{K+1}{\underset{G}{\Rightarrow}} w$ where $w \in T^*$. We need to prove $A \overset{K+1}{\underset{G'}{\Rightarrow}} w$

We may write $w = w_1 w_2 \dots w_n$ and

$$A \underset{G}{\Rightarrow} X_1 X_2 \dots X_n \overset{K}{\underset{G}{\Rightarrow}} w \quad \text{s.t.} \quad X_j \overset{*}{\underset{G}{\Rightarrow}} w_j, \; i \leq j \leq n$$

is in P

- If $X_j \in T$ then $X_j = w_j$
- If $X_j \in V$ then $X_j \overset{*}{\underset{G}{\Rightarrow}} w_j$ implies $X_j \in V'$

by claim (i)

Also $X_j \overset{*}{\underset{G}{\Rightarrow}} w_j$ in at most steps.

Hence by induction hypothesis $X_j \overset{*}{\underset{G'}{\Rightarrow}} w_j$

Also $X_1, X_2 \dots X_n \in (V' \cup T)^*$ implies

$A \to X_1 X_2 \dots X_n$ is in P'

Thus $A \underset{G'}{\Rightarrow} X_1 X_2 \dots X_n \overset{K}{\underset{G'}{\Rightarrow}} w_1 w_2 \dots w_n$

i.e. $A \overset{K+1}{\underset{G'}{\Rightarrow}} w$

In particular, $S \overset{*}{\Rightarrow} w$ for $w \in T^*$

This implies $L(G) \subseteq L(G')$ ——— (ii)

From (i) and (ii), we get $L(G) = L(G')$

Theorem 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 $X \in V' \cup T'$, there exists $\alpha, \beta \in (V' \cup T')^*$ for which

$$S \overset{*}{\Rightarrow} \alpha X \beta$$

Proof:

The set $V' \cup T'$ of symbols appearing in sentential form of G is constructed by an iterative algorithm.

Algorithm 2. Construction 2.

a. Place S in V'

b. If A is placed in V' and $A \to \alpha_1 | \alpha_2 | \ldots | \alpha_n$, then add all variables of $\alpha_1, \alpha_2 \ldots \alpha_n$ to the set V' and all terminals of $\alpha_1, \alpha_2 \ldots \alpha_n$ to T'

c. P' is the set of productions of P containing only symbols of $V' \cup T'$

Example:

Consider $G = (\{S, A, B, E\}, \{a, b, c\}, P, S)$ where P consists of $S \to AB$, $A \to a$, $B \to b$, $E \to c$

Construct an equivalent grammar $G' = (V', T', P', S)$ such that every symbol in $V' \cup T'$ appears in some sentential form of G'

Soln:

0. Initially set $V' = \{S\}$

   $S \to AB$

1. $V' = \{S, A, B\}$.

   $S \to AB$, $A \to a$, $B \to b$

2. $V' = \{S, A, B\}$, $T' = \{a, b\}$

No further modification is possible as $S \to AB$, $A \to a$, $B \to b$.

$\therefore$ $V' = \{S, A, B\}$, $T' = \{a, b\}$

$P' = \{ S \to AB, A \to a, B \to b\}$

Algorithm 1:

input: CFG G with $L(G) \neq \phi$

output: equivalent CFG G' such that every variable in G' derives some terminal string

Algorithm 2:

input: CFG $G = (V, T, P, S)$

output: equivalent CFG $G' = (V', T', P', S)$ such that every symbol in $V' \cup T'$ appears in some sentential form.

Definitions:

- G is said to be reduced or non-redundant if every symbol in $V \cup T$ appears in the course of the derivation of some terminal string

i.e. for every $X \in V \cup T$, $\exists$ some $\alpha, \beta \in (V \cup T)^*$ such that $S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$, where $w \in T^*$

- We say X is useful in the derivation of terminal string.

Theorem 3.

For every CFG G, there exists a reduced grammar G' which is equivalent to G

Proof:

We construct the reduced grammar in two steps:

Algorithm 3 / Construction 3

Step 1:
Construct a CFG G_1 equivalent to a G using Algorithm 1, so that every variable in G_1 derives some terminal string

Step 2:
Construct a CFG G' equivalent to G_1 using Algm.2 so that every symbol (variable as well as terminal symbol) in G' appears in some sentential form of G'

By step 2, every symbol X in G' appears in some sentential form, say $\alpha X \beta$

By step 1, every symbol in $\alpha X \beta$ derives some terminal string.

Note that all symbols of G' are symbols of G

It follows that $S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$ for some terminal string w

Hence G' is reduced.

NOTE:

To get a reduced grammar, we must first apply Algorithm 1 and then Algorithm 2. For; if Algorithm 2 is applied first and then Algorithm 1, we may not get a grammar with its most reduced form

Example:

Find a reduced grammar equivalent to the grammar G whose productions are

$S \to AB \mid CA,\ B \to BC \mid AB,\ A \to a,\ C \to aB \mid b$

Soln:

$G = (V, T, P, S) \xrightarrow[1]{\text{Algm}} G_1 = (V', T', P', S) \xrightarrow[2]{\text{Algm}} G_2 = (V'', T'', P'', S)$

G_1

$W_1 = \{A, C\}$ as $A \to a$, $C \to b$ are productions with a terminal string on right hand side

$W_2 = \{A, C\} \cup S$ as $S \to CA$ is a production with R.H.S. $\in \{T \cup W_1\}^*$

$W_3 = \{A, C, S\} \cup \phi$ as $\{X \in V \mid X \to \alpha \text{ with } \alpha \in (T \cup W_2)^*\}$ is empty

$\therefore V' = \{A, C, S\},\ P' = \{A \to a,\ C \to b,\ S \to CA\}$

G_2

0. $V'' = \{S\}$

   $S \to CA$

1. $V'' = \{S, C, A\}$

   $S \to CA,\ C \to b,\ A \to a$

2. $V'' = \{S, C, A\},\ T'' = \{a, b\}$

   $P'' = \{S \to CA,\ C \to b,\ A \to a\}$

$\therefore$ The reduced equivalent grammar is

$G' = (\{S, A, C\},\ \{a, b\},\ \{S \to CA,\ C \to b,\ A \to a\})$.

Example:

Construct a reduced grammar with no useless symbol equivalent to the grammar

$S \to aAa,\ A \to Sb \mid bCC \mid DaA,\ C \to abb \mid DD,\ E \to aC,$

$D \to aDA$

Soln:

Step 1: $W_1 = \{C\}$ as $C \to abb$ is the only production with a terminal string on the R.H.S.

$W_2 = \{C\} \cup \{A, E\}$ as $A \to bCC$ and $E \to aC$ are productions with R.H.S. in $(T \cup W_1)^*$

$W_3 = W_2 \cup \{S\}$ as $S \to aAa$ and $aAa \in (T \cup W_2)^*$

$W_4 = W_3 \cup \phi$

$\therefore V' = W_4 = \{A, E, C, S\}$

$P' = \{A \to \alpha \mid \alpha \in (V' \cup T)^*\}$

$= \{S \to aAa,\ A \to Sb \mid bCC,\ C \to abb,\ E \to aC\}$

Step 2:

0. $V'' = \{S\}$

   $S \to aAa$

1. $V'' = \{S, A\},\ T'' = \{a\}$

   $S \to aAa,\ A \to Sb \mid bCC$

2. $V'' = \{S, A, C\},\ T'' = \{a, b\}$

   $S \to aAa,\ A \to Sb \mid bCC,\ C \to abb$

3. $V'' = \{S, A, C\},\ T'' = \{a, b\}$

   $P'' = \{S \to aAa,\ A \to Sb \mid bC,\ C \to abb\}$

Algorithm 1: Transforms G to an equivalent G' where every variable derives some terminal strings.

Algorithm 2: Transforms G to an equivalent G' where every symbol appears in some sentential form.

Algorithm 3: Transforms G to an equivalent G' where G' is reduced, i.e. have no useless symbols.

Example:

$S \to aAa,\ A \to Sb \mid bCC \mid DaA,\ C \to abb \mid DD,\ E \to aC$

$D \to aDA$

Algorithm 1:

G': V' $\quad W_0 = \{C\}$

$W_1 = \{C\} \cup \{A, E\} = \{C, A, E\}$

$W_2 = \{A, C, E\} \cup \{S\}$

$= \{S, A, C, E\} = V'$

P' : $S \to aAa,\ A \to Sb \mid bCC,\ C \to abb,\ E \to aC$

Algorithm 2:

G'': $V'' = \{S\},\ S \to aAa$

$V'' = \{S, A\},\ T'' = \{a\},\ A \to Sb \mid bCC$

$V'' = \{S, A, C\},\ T'' = \{a, b\},\ C \to abb$

$V'' = \{S, A, C\},\ T'' = \{a, b\}.$

$P'' = \{S \to aAa,\ A \to Sb \mid bCC,\ C \to abb\}$

Algorithm 4: Transforms G to G' with no null production where $L(G') = L(G) - \{\varepsilon\}$

Algorithm 5: Transforms G to an equivalent G' with no null productions or unit productions

Algorithm 6: Transforms G to an equivalent reduced G' with no null productions or unit productions

Algorithm 4

G $\quad$ $G_1 = (V_1, T_1, P_1, S_1)$

$L(G) = L(G) - \{\varepsilon\}$

↓

G_2 (no null production)

$L(G_2) = L(G)$

↓

G_3 (no unit production)

↓ Algorithm 1

G_4 (all variables derive some terminal string)

↓ Algorithm 2

G_5 (every symbol appears in some sentential form)

- if $\varepsilon \notin L(G)$, then $L(G_1) = L(G)$ set $G_2 = G_1$
- if $\varepsilon \in L(G)$ then $L(G_2) = L(G)$ where

$$G_2 = (V_2, T_2, P_2, S_2), \quad V_2 = V_1 \cup \{S_2\}$$

$$P_2 = P_1 \cup \{S_2 \to \varepsilon \mid S_1\}$$

Reduction to CNF

1. Eliminate null productions and unit productions
2. Elimination of terminals on R.H.S. of each productions
   i. $A \to a$ $A \to BC$ type productions are retained
   ii. $A \to X_1 X_2 \dots X_n$ type productions:
   if some $X_i = a_i$, a terminal, introduce a new variable C_{a_i} and introduce a production $C_{a_i} \to a_i$
3. Restricting the number of variables on R.H.S.

   $A \to A_1 A_2 \dots A_n$, $n \geq 3$ all A_i variables.

   $A \to A_1 C_1$, $C_1 \to A_2 C_2$, $\dots$, $C_{n-1} \to A_{n-1} A_n$

Elimination of Null productions (ε- productions)

- A production of the form $A \to \varepsilon$ is called a null production where A is a variable

A variable A in a CFG is nullable if

$$A \overset{*}{\Rightarrow} \varepsilon$$

Theorem 4

If $L = L(G)$ $(\varepsilon \notin L)$ for some CFG $G = (V, T, P, S)$ then $L - \{\varepsilon\}$ is $L(G')$ for a CFG G' with no null productions

Proof:

We construct $G' = (V, T, P', S)$ as follows:

Construction 4.

We find the nullable variables recursively:

(i) $W_1 = \{A \in V \mid A \to \varepsilon \text{ is in } P\}$

(ii) $W_{i+1} = W_i \cup \{A \in V \mid$ there exists a production $A \to \alpha$ with $\alpha \in W_i^*\}$

Step 1: Construction of the set of nullable variables

By definition, $W_i \subseteq W_{i+1}$ $\forall i$

As V is finite, $W_{k+1} = W_k$ for some $k < |V|$

$\therefore W_{k+j} = W_k$ $\forall j$

Let $W = W_k$

W is set of all nullable variables

i.e. for every variable X in W we have

$$X \overset{*}{\underset{G}{\Rightarrow}} \varepsilon$$

Step 2: Construction of P'

- Any production whose R.H.S. doesnot have any nullable variable is included in P'
- If $A \to X_1 X_2 \ldots X_n$ is in P, then add all productions $A \to \alpha_1 \alpha_2 \ldots \alpha_n$ in P' where
  - i. $\alpha_i = X_i$ if $X_i \notin W$
  - ii. $\alpha_i = X_i$ or ε if $X_i \in W$
  - iii. $\alpha_1 \alpha_2 \ldots \alpha_n = \varepsilon$ i.e. not all α_i's are ε

Thus productions of P' are obtained by

- either not erasing any nullable variables on the R.H.S. of $A \to X_1 X_2 \ldots X_n$ (in P)
- or erasing some or all nullables variables provided some symbol appear on the R.H.S. of $A \to X_1 X_2 \ldots X_n$ after erasing.

Let $G' = (V, T, P', S)$

Claim: G' has no null productions

i.e. $\forall\ A \in V,\ w \in T^*$

$$A \overset{*}{\underset{G'}{\Rightarrow}} w \text{ iff } w \neq \varepsilon \text{ and } A \overset{*}{\underset{G}{\Rightarrow}} w$$

Example:

Consider the grammar G whose productions are

$S \to aS \mid AB,\quad A \to \varepsilon,\quad B \to \varepsilon,\quad D \to b$

Construct a grammar G' without null productions generating $L(G) - \{\varepsilon\}$.

Soln

Step 1: Construction of the set W of all nullable variables

$W_1 = \{A, B\}$

$W_2 = \{W_1 \cup \{A, \in V \mid A_1 \to \alpha \text{ is in } P \text{ with } \alpha \in W_1^*\}$

$= \{A, B, S\}$ as $S \to AB$ is in P with $AB \in W_1^*$

$W_3 = W_2 \cup \phi = W_2$

Thus, $W = \{A, B, S\}$

Step 2: Construction of P'

1. $D \to b$ is included in P'
2. $S \to aS$ gives rise to $S \to aS$ and $S \to a$
3. $S \to AB$ gives rise to $S \to AB$, $S \to A$, $S \to B$

∴ The required CFG without null productions is

$G' = (\{S, A, B\}, \{a, b\}, P', S)$

where P' consists of

$D \to b,\quad S \to aS,\quad S \to a,\quad S \to AB$

$S \to A,\quad S \to B$

To prove $L(G') = L(G) - \{\varepsilon\}$ we prove an auxilliary result given by the following relation

$\forall A \in V$ and $w \in T^*$

$$A \overset{*}{\underset{G'}{\Rightarrow}} w \text{ iff } w \neq \varepsilon \text{ and } A \overset{*}{\underset{G}{\Rightarrow}} w$$

Proof:

(if) Let $A \overset{*}{\underset{G}{\Rightarrow}} w$ and $w \neq \varepsilon$

We prove that $A \overset{*}{\underset{G'}{\Rightarrow}} w$ by induction on the number of steps in the derivation of $A \overset{*}{\underset{G}{\Rightarrow}} w$

Basis:

If $A \underset{G}{\Rightarrow} w$ and $w \neq \varepsilon$ then $A \to w$ is in P' and

so $A \underset{G'}{\Rightarrow} w$

Induction:

Assume the result for derivation in at most i steps.

Let $A \overset{i+1}{\underset{G}{\Rightarrow}} w$ and $w \neq \varepsilon$

We can split the derivation as

$$A \underset{G}{\Rightarrow} X_1 X_2 \cdots X_n \overset{i}{\underset{G}{\Rightarrow}} w_1 w_2 \cdots w_n$$

where

$$w = w_1 w_2 \cdots w_n$$

and $X_j \overset{*}{\underset{G}{\Rightarrow}} w_j$

As $w \neq \varepsilon$, not all w_j's are ε

- if $w_j \neq \varepsilon$, then by induction hypothesis $X_j \overset{*}{\underset{G}{\Rightarrow}} w_j$ and $w_j \neq \varepsilon$ implies $X_j \overset{*}{\underset{G'}{\Rightarrow}} w_j$
- if $w_j = \varepsilon$ then $X_j \in W$ (i.e. X_j is a nullable variable)

  so using production $A \to X_1 X_2 \ldots X_n$ in P we construct production $A \to \alpha_1 \alpha_2 \ldots \alpha_n$ in P' where

$$\alpha_j = \begin{cases} X_j & \text{if } w_j \neq \varepsilon \\ \varepsilon & \text{if } w_j = \varepsilon \text{ (i.e. } X_j \in W) \end{cases}$$

Since $w \neq \varepsilon$ not all α_j's are ε

$$\therefore A \underset{G'}{\Rightarrow} \alpha_1 \alpha_2 \ldots \alpha_n \overset{*}{\underset{G'}{\Rightarrow}} w_1 \alpha_2 \ldots \alpha_n \overset{*}{\underset{G'}{\Rightarrow}} w_1 w_2 \alpha_3 \ldots \alpha_n \overset{*}{\underset{G'}{\Rightarrow}} \vdots \overset{*}{\underset{G'}{\Rightarrow}} w_1 w_2 \ldots w_n = w.$$

(Only if)

We prove 'only if' part by induction on the number of steps in the derivation of $A \overset{*}{\underset{G'}{\Rightarrow}} w$

Basis If $A \underset{G'}{\Rightarrow} w$, then $A \to w$ is in P'

By construction of P', $A \to w$ is obtained from some production $A \to X_1 X_2 \dots X_n$ in P by erasing some (or none of the) nullable variables.

Hence $A \underset{G}{\Rightarrow} X_1 X_2 \dots X_n \overset{*}{\underset{G}{\Rightarrow}} w$.

Induction:

Assume the result for derivation in at most i steps

Let $A \overset{j+1}{\underset{G'}{\Rightarrow}} w$

This can be split as

$$A \underset{G'}{\Rightarrow} X_1 X_2 \dots X_n \overset{*}{\underset{G'}{\Rightarrow}} w_1 w_2 \dots w_n = w$$

where $X_i \overset{*}{\underset{G'}{\Rightarrow}} w_i$ in fewer than $j+1$ steps.

$\therefore$ By induction hypothesis $X_i \overset{*}{\underset{G}{\Rightarrow}} w_i$

The first production $A \to X_1 X_2 \dots X_n$ in P' is obtained from some production $A \to \alpha$ in P by erasing some (or none of the) nullable variables in α

$\therefore A \underset{G}{\Rightarrow} \alpha \overset{*}{\underset{G}{\Rightarrow}} X_1 X_2 \ldots X_n$

- if $X_i \in T$ then $X_i \overset{\circ}{\underset{G}{\Rightarrow}} X_i = w_i$

  if $X_i \in V$ then by induction hypothesis

  $$X_i \overset{*}{\underset{G}{\Rightarrow}} w_i$$

$\therefore A \overset{*}{\underset{G}{\Rightarrow}} X_1 X_2 \ldots X_n \overset{*}{\underset{G}{\Rightarrow}} w_1 w_2 \ldots w_n = w$

Thus by the principle of induction whenever

$A \underset{G'}{\Rightarrow} w$, we have $A \underset{G}{\Rightarrow} w$ and $w \neq \varepsilon$

- Applying the claim to S, we have $w \in L(G')$
  iff $w \in L(G)$ and $w \neq \varepsilon$

  $\Rightarrow L(G') = L(G) - \{\varepsilon\}$

Corollary 1:

An algorithm to decide whether $\varepsilon \in L(G)$ for a CFG:

1. Construct W, the set of the nullable variables.
   at most $|V|$ steps
2. test whether $S \in W$

Corollary 2

If $G = (V, T, P, S)$ is a CFG, we can find an equivalent CFG $G_1 = (V_1, T, P_1, S_1)$ without null productions except $S_1 \to \varepsilon$ when $\varepsilon \in L(G)$. When $S_1 \to \varepsilon$ is in P_1, S_1 doesnot appear on the R.H.S. of any production in P_1.

Proof:

By corollary 1, we can decide whether $\varepsilon \in L(G)$

Case 1:

$\varepsilon \notin L(G)$

Then G_1 obtained by using Theorem 4 is the required equivalent grammar.

Case 2:

$\varepsilon \in L(G)$

Construct $G' = (V, T, P', S)$ using Theorem 4

Then $L(G') = L(G) - \{\varepsilon\}$

Define $G_1 = (V \cup \{S_1\}, T, P_1, S_1)$

where $P_1 = P' \cup \{S_1 \to S, S_1 \to \varepsilon\}$

S_1 doesnot appear on the R.H.S. of any production in P_1

$\therefore$ G_1 is the required CFG with $L(G_1) = L(G)$

Elimination of Unit productions

- A unit production (or a chain rule) in a CFG G is a production of the form $A \to B$ where A and B are variables in G.

Example: G is the grammar: $S \to A$, $A \to B$, $B \to C$, $C \to a$

$S \to A$, $A \to B$, $B \to C$, are useful just to replace S by C

Thus $L(G) = \{a\}$

If G' is the grammar $S \to a$, then $L(G') = L(G)$

Theorem 5

If G is a CFG, we can find a CFG G_1 which has no null productions or unit productions such that

$$L(G_1) = L(G)$$

Proof:

We can apply Corollary 2 of Theorem 4 to grammar G to get a CFG $G' = (V, T, P, S)$ without null productions such that

$$L(G') = L(G)$$

(Algorithm 5 Construction 5)

Let $A \in V$

Step 1: Construction of the set of variables derivable from A

Define $W_i(A)$ recursively as follows:

$$W_0(A) = \{A\}$$

$$W_{i+1}(A) = W_i(A) \cup \{B \in V \mid C \to B \text{ is in } P \text{ with } C \in W_i(A)\}$$

By definition of $W_i(A)$, $W_i(A) \subseteq W_{i+1}(A)$

As V is finite, $W_{K+1}(A) = W_K(A)$ for some $K < |V|$

$\therefore W_{K+j}(A) = W_K(A) \quad \forall\, j \geq 0$

Let $W(A) = W_K(A)$, then $W(A)$ is the set of all variables derivable from A.

Note: $B \in W(A)$ means $A \overset{*}{\underset{G'}{\Rightarrow}} B$

Step 2: Construction of A-productions in G_1

The A-productions in G_1 are

either i. the non-unit A-productions in G'

or ii. $A \to \alpha$ whenever $B \to \alpha$ is in G'

Actually (ii) covers (i) as $A \in W(A)$

Now, we define $G_1 = (V, T, P_1, S)$ where P_1 is constructed using Step 2 for every $A \in V$

Claim: G_1 is the required grammar.

Example:

Let G be $S \to AB, A \to a, B \to c|b, C \to D, D \to E, E \to a$

Eliminate unit productions and get an equivalent grammar.

Soln:

Step 1: $W_0(S) = \{S\}$

$S \to AB$

$W_1(S) = W_0(S) \cup \{A_1 \in V \mid A_2 \to A_1 \text{ is in } P \text{ with } A_2 \in W_0(S)\}$

$= W_0(S) \cup \phi$

$= \{S\}$

$\Rightarrow \boxed{W(S) = \{S\}}$

Similarly $\boxed{W(A) = \{A\}}$

$W_0(B) = \{B\}$

$B \to c|b$

$W_1(B) = W_0(B) \cup \{c\} = \{B, c\}$

$C \to D$

$W_2(B) = W_1(B) \cup \{D\} = \{B, c, D\}$

$D \to E$

$W_3(B) = W_2(B) \cup \{E\} = \{B, c, D, E\}$

$\boxed{W(B) = \{B, c, D, E\}}$

$W_0(c) = \{c\}$

$C \to D$

$W_1(c) = \{c, D\}$

$D \to E$

$W_2(c) = \{c, D, E\}$

$\boxed{W(c) = \{c, D, E\}}$

$W_0(D) = \{D\}$

$D \to E$

$W_1(D) = \{D, E\}$

$\boxed{W(D) = \{D, E\}}$

Step 2:

The productions of G' are

$S \to AB,\ A \to a,\ B \to b,\ E \to a,\ B \to a,\ C \to a,\ D \to a$

(as $\{E \to a$ and $a \notin V\}$ is in G with $E \in W(B)$
$E \in W(C)$
$E \in W(D)$)

Step 3.

To complete the proof, we have to show that

$$L(G') = L(G_1)$$

- $G' = (V, T, P, S)$

  $G_1 = (V, T, P_1, S)$ where consists of

  i. all non-unit productions of G'

  ii. productions $A \to \alpha$ whenever $B \to \alpha$ is in P with $B \in W(A)$ and $\alpha \notin V$

i.e. if $A \to \alpha$ is in $P_1 - P$, then it is induced by $B \to \alpha$ in P with $B \in W(A)$ and $\alpha \notin V$

Now, $B \in W(A)$ implies $A \overset{*}{\underset{G'}{\Rightarrow}} B$

$\therefore\ A \overset{*}{\underset{G'}{\Rightarrow}} B \underset{G'}{\Rightarrow} \alpha$

So, if $A \overset{*}{\underset{G_1}{\Rightarrow}} \alpha$, then $A \overset{*}{\underset{G'}{\Rightarrow}} \alpha$

Thus, $L(G_1) \subseteq L(G')$

To prove $L(G') \subseteq L(G_1)$

Suppose that $w \in L(G')$ and consider a leftmost derivation of w in G', say

$$S \underset{G'}{\Rightarrow} \alpha_1 \underset{G'}{\Rightarrow} \alpha_2 \underset{G'}{\Rightarrow} \cdots \underset{G'}{\Rightarrow} \alpha_n = w$$

Let i be the smallest index such that $\alpha_i \underset{G'}{\Rightarrow} \alpha_{i+1}$ is obtained by a unit production and j be the smallest index greater than i such that $\alpha_j \underset{G'}{\Rightarrow} \alpha_{j+1}$ is obtained by a non-unit production.

So, $S \underset{G_1}{\Rightarrow} \alpha_i$ and $\alpha_j \underset{G'}{\Rightarrow} \alpha_{j+1}$ can be written as

$$\alpha_i = w_i A_i \beta_i \Rightarrow w_i A_{i+1} \beta_i \Rightarrow$$

$$\cdots \Rightarrow w_i A_j \beta_i \Rightarrow w_i \gamma \Rightarrow \alpha_{i+1}$$

when $A_j \in W(A_i)$ and $A_j \to \alpha$ is a non-unit production.

Hence $A_j \to \gamma$ is a production in P_1

$\therefore \alpha_j \overset{*}{\underset{G_1}{\Rightarrow}} \alpha_{j+1}$ and we have $S \overset{*}{\underset{G_1}{\Rightarrow}} \alpha_{j+1}$

Repeating the arguments whenever some unit production occurs in the remaining part of the derivation

$$S \Rightarrow \alpha_1 \Rightarrow \cdots \Rightarrow \alpha_i \Rightarrow \alpha_{i+1} \Rightarrow \cdots \Rightarrow \alpha_j \Rightarrow \alpha_{j+1} \Rightarrow \cdots \Rightarrow \alpha_n = w$$

We can prove that $S \overset{*}{\underset{G_1}{\Rightarrow}} \alpha_n = w$

Hence $L(G') \subseteq L(G_1)$

Corollary

If G is a CFG, we can construct an equivalent grammar G' which is reduced (i.e. has no useless symbols) and has no null productions or unit productions

Proof:

We construct G' in the following way:

Algorithm 6

Eliminate null productions to get G_1

Eliminate unit productions to get G_2

Construction 6

Construct a reduced grammar G' equivalent to G_2

G' is the required equivalent grammar.

Note:

We have to apply the constructions only in the order given above. Otherwise we maynot get the grammar in the most simplified form.