Greibach Normal Form (GNF)

In this form, we put restrictions on the positions in which terminal & variables can appear & not on the length of the right sides of a production (as in CNF)

Def^n. A CFG is said to be in GNF if all productions have the form $A \rightarrow ax$, where

$a \in T$ and $x \in V^*$.

Case I:- If starting symbol is Variable (Non-Terminal).

eg 1. Convert
$$S \rightarrow AB,$$
$$A \rightarrow aA \mid bB \mid b,$$
$$B \rightarrow b$$
in GNF.

sol^n:
$$S \rightarrow aAB \mid bBB \mid bB$$
$$A \rightarrow aA \mid bB \mid b$$
$$B \rightarrow b.$$

case II:- If starting symbol is terminal.

eg 2 $S \rightarrow abSb \mid aa$

sol^n:-
$$S \rightarrow aBSB \mid aA$$
$$A \rightarrow a$$
$$B \rightarrow b$$

eg:- Convert $S \to ABa$
$A \to aab$
$B \to Ac$ into CNF.

Sol^n.

Ist step :-

$S \to A B B_a$,
$A \to B_a B_a B_b$,
$B \to A B_c$,
$B_a \to a$,
$B_b \to b$,
$B_c \to c$

IInd step :-

$S \to A D_1$,
$D_1 \to B B_a$,
$A \to B_a D_2$,
$D_2 \to B_a B_b$,
$B \to A B_c$,
$B_a \to a$,
$B_b \to b$,
$B_c \to c$

where $C_i = x_i$ if x_i is in V,

and $C_i = B_a$ if $x_i = a$

For every B_a, we also put into P, the production

$B_a \rightarrow a$.

This part of the algorithm removes all terminals from productions whose right side has length greater than one, replacing them with newly introduced variables.

At the end of this step, we have a grammar G_1 all of whose productions have the form

$A \rightarrow a$ (2) or $A \rightarrow C_1 C_2 \cdots C_n$ where $C_i \in V_1$. (3)

Thus, $L(G_1) = L(G)$.

<u>Step II.</u> In second step, we introduce additional variables to reduce the length of right sides of the productions where necessary.

First, we put all production of the form (2) & (3) with $n = 2$ into $\hat{P}$.

For $n > 2$, we introduce new variables $D_1, D_2, \ldots$ & put into $\hat{P}$ the productions

$$\begin{aligned} A &\rightarrow C_1 D_1 \\ D_1 &\rightarrow C_2 D_2 \\ &\vdots \\ D_{n-2} &\rightarrow C_{n-1} C_n. \end{aligned}$$

$\therefore L(\hat{G}) = L(G)$ as $\hat{G}$ is in CNF.

6.2. Two important Normal Forms.

Chomsky Normal Form (CNF)

A CFG is in CNF if all productions are of the form

$A \to BC$ or $A \to a$ where

A, B, C are in V & 'a' is in T.

Th 6.6. Any CFG $G = (V, T, S, P)$ with $\lambda \notin L(G)$ has an equivalent grammar $\hat{G} = (\hat{V}, \hat{T}, S, \hat{P})$ in CNF.

Proof:- Because of Previous Theorems (6.5), we assume that G has no λ- productions & no unit- productions.

The construction of $\hat{G}$ will be done in two steps:-

step 1: Construct a grammar $G_1 = (V_1, T, S, P_1)$ from G by considering all productions in P in the form

$$A \to x_1 x_2 \cdots x_n, \quad \text{—(1)}$$

where each x_i is a symbol either in V or in T.

If $n = 1$, then x_1 must be a terminal since we have no unit- productions. ∴ Put the production is P_1.

If $n \geq 2$, introduce new variable B_a for each $a \in T$.

For each production of P in the form (1), we put into P_1 the production $A \to C_1 C_2 \cdots C_n$, Th 6.5: Let L be a CFL that does not contain λ. Then there exists a CFG that generates L & that does not have any useless productions, λ-production, or unit productions.

Proof:- The procedures given in Th 6.2, 6.3 & 6.4 remove these kinds of production in turn. The only point that needs consideration is that the removal of one type of production may introduce productions of another type. For eg, the procedure for removing λ-productions can create new unit-productions. But removal of unit-productions does not create λ-productions. & removal of useless productions does not create λ-productions or unit-productions. Therefore, we can remove all undesirable productions using the following seq. of steps:

1. Remove λ-productions.
2. Remove unit-productions
3. Remove useless productions.

eg:- Remove all unit-Productions from,

$S \to Aa \mid B,$

$B \to A \mid bb,$

$A \to a \mid bc \mid B.$

Sol^n. The dependency graph for the unit-Productions is

we see that, $S \overset{*}{\Rightarrow} A$, $S \overset{*}{\Rightarrow} B$, $B \overset{*}{\Rightarrow} A$, $A \overset{*}{\Rightarrow} B$

are all the unit productions in given CFG.

Hence, we add to the original non-unit productions.

$S \to Aa$

$B \to bb$

$A \to a \mid bc$, the new Rules,

$S \to a \mid bc \mid bb,$

$A \to bb,$

$B \to a \mid bc.$

to obtain the equivalent grammar

$S \to a \mid bc \mid bb \mid Aa,$

$A \to a \mid bc \mid bb,$

$B \to a \mid bb \mid bc.$

\# Removal of unit Productions has made 'B' and associated productions useless.

Scanned by CamScanner

f:- Any unit-Production of the form $A \to A$ can be removed from the grammar without effect (does not generate any terminal), & thus we consider $A \to B$, where A & B are different variables.

Suppose, $A \to B$ &

$B \to y_1 | y_2 | \cdots | y_n$ then removing the unit production (applying substitution Rule) it becomes,

$$A \to y_1 | y_2 | \cdots | y_n .$$

But, this will not always work, suppose,

$A \to B$, $B \to A$. , the unit productions are not removed. Sol^n to this is, first find for A, all variable B such that

$$A \overset{*}{\Rightarrow} B \quad \text{—} (1)$$

Do this, by drawing a dependency graph with an edge (C, D) whenever the grammar has a unit-production $C \to D$ then (1) holds whenever there is a walk b/w A & B. The new grammar $\hat{G}$ is generated by first putting into $\hat{P}$ all non-unit productions of P. Next for all A & B satisfying (1), we add to $\hat{P}$

$$A \to y_1 | y_2 | \cdots | y_n, \text{ where}$$

$B \to y_1 | y_2 | \cdots | y_n$ is the set of all rules in $\hat{P}$ with B on the left. Since $B \to y_1 | y_2 | \cdots | y_n$ is taken from $\hat{P}$, none of the y_i can be a single variable, so that no unit-Productions are created by the last step. $\therefore L(G) = L(\hat{G})$.

Scanned by CamScanner

eg:- find CFG without λ- Productions equivalent to grammar defined by

$$S \to ABaC$$
$$A \to BC$$
$$B \to b \mid \lambda$$
$$C \to D \mid \lambda$$
$$D \to d$$

Solⁿ: First find nullable variables that are: A, B, C., then the resulting CFG is,

$$S \to ABaC \mid BaC \mid AaC \mid ABa \mid ac \mid Ba \mid Aa \mid a$$
$$A \to BC \mid B \mid C$$
$$B \to b$$
$$C \to D$$
$$D \to d$$

(IV) Removing Unit - Productions

Any production of a CFG of the form $A \to B$, where $A, B \in V$, is called a unit production.

Th 6.4: Let $G = (V, T, S, P)$ be any CFG without λ-Productions. Then there exists a CFG $\hat{G} = (\hat{V}, \hat{T}, S, \hat{P})$ that does not have any unit-productions & that is equivalent to G.

6.3.

Let G be any CFG with λ not in $L(G)$. Then there exits an equivalent grammar $\hat{G}$ having no λ-productions.

Proof:- First, find the set V_N of all nullable variables of G, using the steps:-

1) for all productions $A \rightarrow \lambda$, Put A into V_N.

2) Repeat foll. step until no further variables are added to V_N.

for all productions,

$$B \rightarrow A_1 A_2 \dots A_N,$$

where all $A_1, A_2, \dots A_n$ are in V_N, Put B into V_N.

set V_N is found & now construct $\hat{P}$. for this, we look at all the productions in P of the form,

$$A \rightarrow x_1 x_2 \dots x_m, \; m \geq 1, \text{ where each } x_i \in V \cup T.$$

for each such production of P, we put into $\hat{P}$ that production as well as those generated by replacing nullable variables with λ in all possible combinations.

Thus, the grammar $\hat{G}$ is equivalent to G as $L(G) = L(\hat{G})$.

(III) Removing λ - Productions.

Any Production of a CFG of the form $A \rightarrow \lambda$ is called a λ - Production. Any variable A for which the derivation $A \overset{*}{\Rightarrow} \lambda$ is possible is called nullable.

A Grammar may generate a language not containing λ, yet have some λ - productions or nullable variables. In such cases, λ-productions can be removed.

eg:-

$S \rightarrow aS_1b$

$S_1 \rightarrow aS_1b \mid \lambda$

soln:-

$S \rightarrow aS_1b \mid ab$

$S_1 \rightarrow aS_1b \mid ab$

The null production $S_1 \rightarrow \lambda$ can be removed after adding new productions obtained by substituting λ for S_1 where it occurs on the right.

Th 6.2. Let $G = (V, T, S, P)$ can be a CFG, Then there exists an equivalent grammar $\hat{G} = (\hat{V}, \hat{T}, S, \hat{P})$ that does not contain any useless variables or productions.

Proof. The Grammar $\hat{G}$ can be generated from G by an algo. consisting of two parts.

I st Part :- we construct an intermediate grammar $G_1 = (V_1, T_2, S, P_1)$ such that V_1 contains only variables A for which

$A \overset{*}{\Rightarrow} w \in T^*$ is possible. The steps in algo are

1. Set V_1 to ϕ
2. Repeat the foll. step until no more variables are added to V_1. For every $A \in V$ for which P has a production of the form $A \to x_1 x_2 \dots x_n$, with all x_i in $V_1 \cup T$,

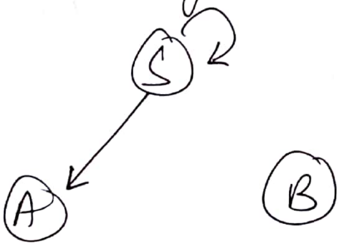
   add A to V_1.
3. Take P_1 as all the productions in P whose symbols are all in $(V_1 \cup T)$.

II nd Part: Draw variable Dependency graph for G_1 & from it find all variables that cannot be reached from S. These are removed from the variable set. Also eliminate any terminal that does not occur in some useful production. Result is $\hat{G} = (\hat{V}, \hat{T}, S, \hat{P})$. we have both the derivations

$S \overset{*}{\Rightarrow} xAy \overset{*}{\Rightarrow} w$ and $S \overset{*}{\Rightarrow}_{\hat{G}} xAy \overset{*}{\Rightarrow}_{\hat{G}} w$.

Thus G & $\hat{G}$ are equivalent.

Dependency Graph.



Here B is useless. Removing 'B' & productions of 'B', we get,

$\hat{G} = (\hat{V}, \hat{T}, S, \hat{P})$ with $\hat{V} = \{S, A\}$,
$\hat{T} = \{a\}$ &

Productions :-

$S \to aS \mid A$

$A \to a$.

Q: Eliminate useless symbols & productions from $G = (V, T, S, P)$ where $V = \{S, A, B, C\}$ & $T = \{a, b\}$ with P consisting of

$S \to aS \mid A \mid C,$

$A \to a$

$B \to aa,$

$C \to aCb.$

Solⁿ: first identify variables that lead to terminal string.

$A \to a$

$B \to aa$

$S \to aS \Rightarrow aA \Rightarrow aa.$

Thus, # C cannot lead to terminal string.

grammar G_1 has variables $V_1 = \{S, A, B\}$ &

$T_1 = \{a\}$ &

Productions $S \to aS \mid A$

$A \to a$

$B \to aa$

Secondly eliminate variables that cannot be reached from starting variable. Draw Dependency graph with vertices as variables edge b/w C & D only if there is a production of the form

$C \to xDy$

$G = (V, T, S, P)$
(INT), (t), (start Prod), (set of Prod's)

Eg: Let $G = (\{A, B\}, \{a, b, c\}, A, P)$ with Productions

$A \rightarrow a \mid aaA \mid abBc,$

$B \rightarrow abbA \mid b$

use Substitution Rule to get the Grammar $\hat{G}$ such that $L(G) = L(\hat{G})$.

Sol^n:

$A \rightarrow a \mid aaA \mid ab\underline{abbA}c \mid ab\underline{b}c$

$B \rightarrow abbA \mid b$

variable 'B' & its productions are still in the grammar even though they no longer play a part in any derivation.

II. Removing useless productions

Reasons why a variable is useless

1. If that variable cannot be reached from start symbol. (a variable is useful iff it occurs in atleast one derivation)
2. If it cannot derive a terminal string.

eg: of Reason 1

$S \rightarrow A$

$A \rightarrow aA \mid \lambda$

$B \rightarrow bA$

Variable B is useless & so is the production $B \rightarrow bA$. Although B can derive a terminal string, there is no way we can achieve $S \overset{*}{\Rightarrow} xBy$.

Now, suppose we start with the starting production S, we get,

$$S \overset{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2$$ [u_1, u_2 any arb. terminals & $1 \leq j \leq n$]

using second grammar $\hat{G}$, we get,

$$S \overset{*}{\Rightarrow}_{\hat{G}} u_1 A u_2 \Rightarrow_{\hat{G}} u_1 x_1 y_j x_2 u_2$$

Thus same sentinel can be reached from G & $\hat{G}$.

If ① is used again later, we can repeat the argument. It follows then, by induction on the no. of times the production is applied, that $S \overset{*}{\Rightarrow}_{\hat{G}} w$.

Therefore, if $w \in L(G)$, then $w \in L(\hat{G})$.

By similar reasoning, we can show that if $w \in L(\hat{G})$, then $w \in L(G)$.

P. Linz - An Int. to formal Languages & Automata
-4th Ed. (6.1, 6.2) in course.

6.1. Methods for transforming Grammars.

I) Substitution Rule.

Th 6.1. Let $G = (V, T, S, P)$ be a CFG.

V - variables / Non Ter.
T - terminals
S → start of production NT.
P → Productions.

suppose that P contains a production of the form $A \to x_1 B x_2$ &

$$B \to y_1 | y_2 | \dots | y_n \quad \text{— (1)}$$

Let $\hat{G} = (V, T, S, \hat{P})$ be a grammar in which $\hat{P}$ is constructed by deleting $A \to x_1 B x_2$ from P & adding to it,

$$A \to x_1 y_1 x_2 \,|\, x_1 y_2 x_2 \,|\, \dots \,|\, x_1 y_n x_2 . \quad \text{— (2)}$$

Then $L(\hat{G}) = L(G)$.

Proof :- for eg :- $G = \begin{cases} S \to aXa \\ X \to b \end{cases}$ $\hat{G} = \begin{cases} S \to aba \end{cases}$

$\therefore w = aba.$

suppose that $w \in L(G)$, so that $S \overset{*}{\Rightarrow}_G w$. [ie w is derived from grammar G]

If the derivation of w [for eg :- aba] does not include (1) that means the word is derived from second grammar ie $S \overset{*}{\Rightarrow}_{\hat{G}} w$.

Scanned by CamScanner