CHAPTER - 7

# HW - FCS

## Submission Number # 04

Kiran Shettar
UML ID - 01605800

F.1.3

$S \rightarrow 0A0 \mid 1B1 \mid BB$

$A \rightarrow C$

$B \rightarrow S \mid A$

$C \rightarrow S \mid \epsilon$

Eliminating $\epsilon$ - production

Stage - I

| Old NT | Production | New NT |
|---|---|---|
| | $C \rightarrow \epsilon$ | C |
| $\phi$ | $A \rightarrow C$ | C, A |
| C | $B \rightarrow A$ | C, A, B |
| C, A | $S \rightarrow BB$ | C, A, B, S |
| C, A, B | $B \rightarrow S$ | C, A, B, S |
| C, A, B, S | $C \rightarrow S$ | |

Stage - II

$S \rightarrow 0A0 \mid 00$

$S \rightarrow 1B1$

$S \rightarrow BB \mid B$

$A \rightarrow C$

$B \rightarrow S \mid A$

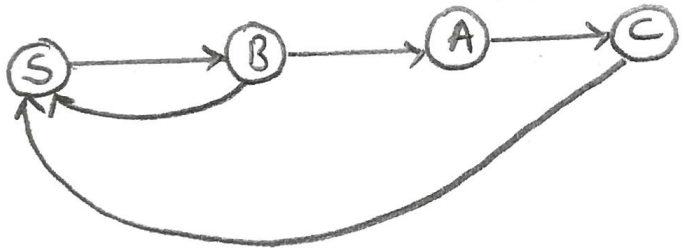
$C \rightarrow S \mid \epsilon$

Unit production:

Stage I

| Unit Production | Non-unit production. |
|---|---|
| $S \rightarrow B$ | $S \rightarrow OAO \mid 00$ |
| $A \rightarrow C$ | $S \rightarrow 1B1 \mid 11$ |
| $B \rightarrow S$ | $S \rightarrow BB$ |
| $B \rightarrow A$ | |
| $C \rightarrow S$ | |

Stage II




CNF

$S \rightarrow 00 \mid 11 \mid OAO \mid 1B1 \mid BB$

$A \rightarrow 00 \mid 11 \mid OAO \mid 1B1 \mid BB$

$B \rightarrow 00 \mid 11 \mid OAO \mid 1B1 \mid BB$

$S \rightarrow G_1 G_1$

$S \rightarrow G_2 G_2$

$S \rightarrow OAO \Rightarrow S \rightarrow (G_1 A) G_1 \Rightarrow S \rightarrow G_3 G_1$

$S \rightarrow 1B1 \Rightarrow S \rightarrow (G_2 B) G_2 \Rightarrow S \rightarrow G_4 G_2$

$S \rightarrow BB$

$A \rightarrow G_1 G_1$

$A \rightarrow G_2 G_2$

$A \rightarrow G_3 G_1$

$A \rightarrow G_4 G_2$

$A \rightarrow BB$

$B \rightarrow G_1 G_1$

$B \rightarrow G_2 G_2$

$B \rightarrow G_3 G_1$

$B \rightarrow G_4 G_2$

$B \rightarrow BB$

$G_1 \rightarrow 0$

$G_2 \rightarrow 1$

$G_3 \rightarrow G_1 A$

$G_4 \rightarrow G_2 B$

CNF is:

$S \rightarrow G_1 G_1 \mid G_2 G_2 \mid G_3 G_1 \mid G_4 G_2 \mid BB$

$A \rightarrow G_1 G_1 \mid G_2 G_2 \mid G_3 G_1 \mid G_4 G_2 \mid BB$

$B \rightarrow G_1 G_1 \mid G_2 G_2 \mid G_3 G_1 \mid G_4 G_2 \mid BB$

$G_1 \rightarrow 0$

$G_2 \rightarrow 1$

$G_3 \rightarrow G_1 A$

$G_4 \rightarrow G_2 B$

Stage III :

$C \rightarrow 0A0 \mid 00 \mid 1B1 \mid 11 \mid BB$

$A \rightarrow 0A0 \mid 00 \mid 1B1 \mid 11 \mid BB$

$B \rightarrow 0A0 \mid 00 \mid 1B1 \mid 11 \mid BB$

$S \rightarrow 0A0 \mid 00 \mid 1B1 \mid 11 \mid BB$

Useless Production:

Stage -I :

| Old NT | Production | New NT |
|---|---|---|
| $\phi$ | $C \to 00 \mid 11$<br>$A \to 00 \mid 11$<br>$B \to 00 \mid 11$<br>$S \to 00 \mid 11$ | C, A, B, S |
| C, A, B, S | $C \to 0A0 \mid 1B1 \mid BB$<br>$A \to 0A0 \mid 1B1 \mid BB$<br>$B \to 0A0 \mid 1B1 \mid BB$<br>$S \to 0A0 \mid 1B1 \mid BB$ | C, A, B, S. |

Stage -II :

| Production | Terminal | Non-terminal |
|---|---|---|
| — | — | S |
| $S \to 00 \mid 11$ | 0, 1 | A, B |
| $S \to 0A0 \mid 1B1 \mid BB$ | | |
| $A \to 00 \mid 11$ | 0, 1 | A, B. |
| $A \to 0A0 \mid 1B1 \mid BB$ | | |
| $B \to 00 \mid 11$ | | |
| $B \to 0A0 \mid 1B1 \mid BB$ | | |

F.1.4

$S \rightarrow AAA \mid B$

$A \rightarrow aA \mid B$

$B \rightarrow \epsilon$

Eliminating $\epsilon$-productions:

$S \rightarrow AAA \mid AA \mid A$, $S \rightarrow B$

$A \rightarrow aA \mid a$

$A \rightarrow B$

Eliminating unit productions:

Stage-I:

$S \rightarrow A$

$S \rightarrow B$

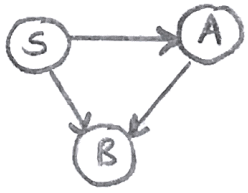
$A \rightarrow B$

~~NEDB~~ NUP

$S \rightarrow AAA \mid AA$

$A \rightarrow aA \mid a$

Stage II:




Stage III:

$A \rightarrow aA \mid a$

$S \rightarrow AAA \mid AA \mid aA \mid a$

Eliminating useless production:

Stage I

| Old NT | Production | New NT |
|---|---|---|
| | A | A, S |
| ϕ | S | |
| A, S | A → aA | |
| | S → AAA | A, S |
| | S → AA | |
| | S → aA | |

Stage II

| Production | Terminal | Non-Terminal |
|---|---|---|
| – | – | S |
| S → a | a | A |
| S → AAA | | |
| S → AA \| aA | a | A |
| A → a \| aA | | |

CNF:

S → a
S → AS₁
S₁ → AA
S → AA | A₁A
A → a | A₁A
A₁ → a

7.1.5 $S \to aAa \mid bBb \mid \epsilon$

$A \to C \mid a$

$B \to C \mid b$

$C \to CDE \mid \epsilon$

$D \to A \mid B \mid ab$

ε- productions:

Stage I:

| Old NT | Production | New NT |
|---|---|---|
| ∅ | $S \to \varepsilon$ | S, C |
| S, C | $C \to \varepsilon$<br>$A \to C$<br>$B \to C$ | S, C, A, B |
| S, C, A, B | $D \to A$<br>$D \to B$ | S, C, A, B, D |
| S, C, A, B, D | – | S, C, A, B, D |

Stage II:

$S \to aAa \mid aa$

$S \to bBb \mid bb$

$A \to C \mid a$

$B \to C \mid b$

$C \to CDE \mid DE \mid CE \mid E$

$D \to A \mid B \mid ab$

Unit productions:

Stage I

Unit productions

$A \to C$

$B \to C$

$CB \to E$

$D \to A \mid B$

Non-unit production

$S \to aAa \mid aa$

$S \to bBb \mid bb$

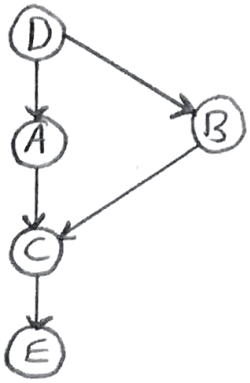
$A \to a$

$B \to b$

$C \to CDE \mid DE \mid CE$

$D \to ab$

Stage II




Stage III

$C \to CDE \mid DE \mid CE$

$A \to a \mid CDE \mid DE \mid CE$

$D \to ab \mid a \mid CDE \mid DE \mid CE \mid b$

$B \to b \mid CDE \mid DE \mid CE$

$S \to aAa \mid aa \mid bBb \mid bb$

Useless productions:

Stage I

| old NT | Production | New NT |
|---|---|---|
| $\phi$ | $A \to a$ | |
| | $D \to b$ | A, B, D, S |
| | $D \to ab \mid a$ | |
| | $B \to b$ | |
| | $S \to aa \mid bb$ | |
| A, B, D, S | $S \to aAa$ | A, B, D, S |
| | $S \to bBb$ | |

Stage II

| Production | Terminal | Non-terminal. |
|---|---|---|
| — | — | S |
| $S \to aa \mid bb$ | a, b | A, B |
| $S \to aAa$ | | |
| $S \to bBb$ | | |
| $A \to a$ | a, b | A, B |
| $B \to b$ | | |

CNF:

$S \to aa \mid bb \mid aAa \mid bBb$

$A \to a$

$B \to a$

$S \to A_1 A_1$

$S \to B_1 B_1$

$S \to C_1 A_1$

$S \to C_2 B_1$

$A \to a$

$B \to b$

$A_1 \to a$

$B_1 \to b$

$S \to AA$

$S \to BB$

$S \to AAA \Rightarrow S \to A_1 A$

$S \to BBB \Rightarrow S \to B_1 B$

$A \to a$

$B \to b$

$A_1 \to AA$

$B_1 \to BB$

$\therefore S \to AA \mid BB \mid A_1 A \mid B_1 B$

$A \to a$

$B \to b$

$A_1 \to AA$

$B_1 \to BB.$

T.1.6)

<[({})]>

S → AB | CD | EF | GH

A → {

B → }

C → (

D → )

E → [

F → ]

G → <

H → >

T.1.7) ~~Let 'A' be the symbol of non terminals~~ Let 'A' be the start symbol of the grammar with non-terminals $A_1, A_2, A_3 \ldots A_p$ where $A = A_1$

Let us assume the productions $A_1, A_2 \ldots A_{p-1}$ produces 'n' non terminals & $A_p$ produces $\varepsilon$.

∴ The grammar looks like

$A_1 \rightarrow A_2, A_2 \ldots A_2$

$A_2 \rightarrow A_3, A_3 \ldots A_3$

⋮

$A_{p-1} \rightarrow A_p A_p \ldots A_p$

$A_p \rightarrow \underbrace{\varepsilon \ldots \varepsilon}_{n}$

In the grammar we see that each of the $n A_1$ non-terminals produced by $A_1$ is replaced by 'n', additional non-terminal $A_2$ and so on.

Thus the parse tree would have $A_1$ single root, 'n' non terminals in second level, $n^2$ non-terminals in third level, thereby the last row of internal nodes contains $n^{p-1}$ non-terminal. Thereby, yeilding geometric series in the number of non-terminals replaced:

$$\sum_{k=0}^{p-1} n^k = 1 + n + n^2 + \dots + n^{p-2} + n^{p-1}$$

$$= \frac{(n^{p-1+1} - 1)}{(n-1)} = \frac{(n^p - 1)}{(n-1)}$$

If the grammar is extreme case $L = \{\varepsilon\}$ then the bound is actually reached.

**T.1.8** The statement of the entire construction may be a bit tricky, since we need to use the construction of part in the construction for (b) is by induction on i, but it needs to be of the stronger statement, that if an $A_i$ production has a body beginning with $A_j$, then $j > i$ (i.e we use part (c) to eliminate the possibility that $i = j$).

Basis: for $i=1$ we simply apply the construction of (c) for $i=1$

Induction: If there's any production of the form $A_i \to A_1 \ldots$, use the construction of (a) to replace $A_1$. That gives us a situation where all $A_i$ production bodies begin with atleast $A_2$ or a terminal.

Similarly replace initial $A_2$'s using (a), to make $A_3$ the lowest possible variable beginning an $A_i$ production. In this manner, we eventually guarantee that the body of each $A_i$ production either begins with a terminal or $A_j$, for some $j >= i$. A use of the CNF construction eliminates the possibility that $i=j$ and increases the runtime to $n^2$. Hence, it can be seen that CNF grammar has at most $O(n^2)$ productions.

7.1.8 (b) Similar to the above proof (a) it can be shown that CNF is proportional to $n^2$ productions.

Basis: for $i=1$ we simply apply the construction of (c) for $i=1$.

Induction: If there's any production of the form $A_i \to A1 \ldots$, use the construction of (a) to replace $A1$. that gives a situation where all $A_i$ productio bodies begin with atleast $A_2$ or a terminal.

Similarly, replace initial $A2$'s using (a), to make $A3$ the lowest possible variable beginning an $A_i$-production. In this manner we eventually guarantee that the body of each $A_i$ production begins with a terminal or with $A_j$, for some $j \geq i$ A use of the CNF construction eliminates the possibili -ty that $i = j$ & increases the run time to $n^2$ & hence proved this.

T.1.9 a) Induction: Assume the statement for indexes greater than $i$. If an $A_i$-production begins with a variable, it must be $A_j$ for some $j > i$. By the induction hypothesis, the $A_j$ productions all have bodies beginning with terminals now. Thus we may use the construction (a) to replace the initial $A_j$, yeilding only $A_i$ productions whose bodies begin with terminals.

7.1.9 b) After fixing all the $A_i$ productions for all $i$, it's time to work on the $b_i$-productions. Since these have bodies that begin with either terminals or $A_j$ for some $j$, and the latter variables have only bodies that begin with terminals, application of construction (a) fixes the $B_j$'s.

7.1.9 c) All are unit pairs, As per the hint, we do a backwards induction on $i$, that the bodies of $A_i$ productions can be made to begin with terminals

Basis: for $i = k$ there ~~exists~~ is nothing to do, since there're no variables with index higher than 'k' to begin the body.

Induction: Assume the statements for indexes greater than $i$. If an $A_i$-production begins with a variable, it must be $A_j$ for some $j > i$

By induction hypothesis, the $A_j$ productions all have bodies beginning with terminals now. Thus we may use the construction (a) to replace the initial $A_j$, yeilding only $A_i$ productions whose bodies begin with terminals.

T.1.11

b) The statement for the entire construction is a bit confusing, as we need to use the construction of part (c) in (b), although part (c) doesn't use part(b) solution in any manner.

The construction for (b) is by induction on 'i', but it needs to be of the stronger statement that if an $A_i$-production has a body beginning with $A_j$ then $j > i$ (i.e we use part (c) to eliminate the possibility that $i = j$).

Basis: For $i = 1$, we simply apply the construction of (c) for $i = 1$.

Induction: If there is any production of the form $A_i \to A_1 \ldots$, use the construction of (a) to replace $A_1$ That gives us a situation where all $A_i$ production bodies begin with atleast $A_2$ or a terminal. Similarly replace initial $A_2$'s using (a) to make $A_3$ the lowest possible variable beginning an $A_i$ production. In this manner, we eventually tell that the body of each $A_i$ production either begin with a terminal with $A_i$, for some $j \geq i$, A use of construction from (c) eliminates the possibility that $i = j$.

7.1.11(d) As per the hint, we do a backward induction on 'i', that the bodies of $A_i$ productions can be made to begin with terminals.

Basis: For $i = k$, there's nothing to do, since there're no variables with index higher than 'k' to begin the body.

Induction: Assume the statement for indices greater than 'i'. If an $A_i$ production begins with a variable, it must be $A_j$ for some terminals $j > i$. By induction hypothesis, $A_j$ productions have the bodies beginning with terminals now. Thus we may use construction (a) to replace the initial $A_i$, yeilding only $A_i$ productions whose bodies begin with terminals. After fixing all the $A_i$ productions for all i, it's time to work on the $B_i$ productions. Since these have bodies that begin with terminals, application of construction (a) fixes the $B_j$'s.

7.2.1 b) Let us use the language $\{a^n b^n c^i \mid i \leq n\}$.

for any constant greater that 0, take a string to be $z = a^n b^n c^n$. Clearly $z \in L$. Now the string decomposes into $z = uvwxy$, with $vwx \neq \epsilon$ and $|vwx| \leq n$. We then have several cases to consider.

→ $vwx \in a^+$

Pump up & we will have a's more than b's. This doesn't belong to 'L'.

→ $vwx \in b^+$

Pump up & we will have b's more than a's. This doesn't belong to 'L'.

→ $vwx \in c^+$

Pump up & we will have c's more than a's & b's. This doesn't belong to L.

→ $vwx \in a^+ b^+$

Pump down & we will have less a's & less b's than c's. This doesn't belong to L.

→ $vwx \in b^+ c^+$

Pump up & will have more c's than a's

This doesn't belong to L.

7.2.1 c) Let $\Sigma = a$ $p = n$

$L = \{ a^n : n \text{ is prime} \}$

Assume for contradiction that L is a CFL

We apply pumping lemma.

Let 'm' be a parameter for pumping lemma. Let 'p' be a prime such that $p \geq m$. We choose to pump the string $a^p \in L$. Since $a^p = uvxyz$, we have that $v = a^k$ & $y = a^l$, with $k + l \geq 1$ (since $|vy| \geq 1$). From the pumping lemma we have that $uv^i xy^i z \in L$ and therefore $a^{p+kp+lp} \in L$.

Subsequently $a^{p(1+k+l)} \in L$, which is impossible since $p(1+k+l)$ is not a prime.

Thus we have a contradiction & the language 'L' is not context free.

7.2.1 e) Assume for a sake of deriving a contradiction that $L = \{ a^n b^n c^i \mid n \leq i \leq 2n \}$ is context free.

This implies that there exists some length 'k' such that all string 'z', where

$|z| \geq k$ can be decomposed so that $z' = uvwxy$ where $|vwx| \leq k$ and $vx \neq \varepsilon$, and for any $i \geq 0$, the string $z' = uv^i wx^i y$ should also be in 'L'.

Choose $z = a^k b^k c^{2k}$. This string is in 'L' because the number of a's is equal to the number of b's, and the number of c's in the string is obviously less than or equal to the number of a's, and b's. Also $|z| = 4k$ is greater than or equal to 'k'. Because $|vwx| \leq k$, vwx is either contained entirely by the a's, b's or c's. We will consider each case separately.

* $vwx = a^m$ where $a \in \{a, b, c\}$ and $m \leq k$

If $\sigma = a$ or $\sigma = b$ pumping down so that $z' = uv^0 wx^0 y$ produces a string that is not in L. Suppose $\sigma = a$, because $|vx| > 0$, $z' = a^{k-|vx|} b^k c^k$, which is not in 'L' because the number of a's doesn't equal the number of b's. Then reasoning when $\sigma = b$ is symmetric

If $\sigma = c$, pumping up so that $z' = uv^2 wx^2 y$ produces a string that's not in 'L'. Because $|vx| > 0$, $z' = a^k b^k c^{2k+|vx|}$. This string is not in L

because, the number of c's is greater than twice the number of a's & b's

* $vwx = \sigma^l \gamma^m$, where $\sigma, \gamma \in \{a, b, c\}$, $\sigma \neq \gamma$ and $l + m \leq k$.

If $\sigma = a$ & $\gamma = b$, no matter how the symbols are partitioned into v & x, we can select an appropriate 'i' so that pumping up by 'i' results in the number of a's & b's being greater than the number of c's. Selecting $i = 2k+1$, so that the length of left substring is $2k + 2k|v| + 2k|x|$ results in $3' \notin L$, because at least one of $|v|$ or $|x|$ is greater than zero.

If $\sigma = b$ & $\gamma = c$. if either v or x contains c's, we can select i so that pumping up results in the number of c's being greater than twice the number of a's or b's and the string is not in L. If neither v's nor x contains c's, then pumping down results in the number of b's being fewer than number of a's and again $3' \notin L$.

for all possible partitions of 3, there exists some i such that $3' = uv^i wx^i y \notin L$.

This cannot be true given our assumption that 'L' is context free, so 'L' must not be context free.

T.2.1 > f> Let 'n' be the pumping lemma constan and consider the string $3 = ww^Rw$. We may write $3 = ww^Rw$, where v and x may be pumped and $|vw^R| \leq n$. If vwx doesn't have c's, then $uv^3wx^3y$ has atleast $n+2$ a's or b's, and thus could not be in the language. If vwx has a 'c', then it could not have an 'a', bcz it's length is limited to 'n'. Thus uvy has n a's, but more than $2n+2$ b's and c's in total. Thus it's not possible that uwy has more b's than a's and also has more c's than b's.

We conclude that uwy is not in the language, and now have a contradiction no matter how '3' is broken into $3 = ww^Rw$.

T.2.2. a> Proving a language is not context-free using pumping lemma goes like this

a> {00, 11}

Let 'n' be a constant as in the pumping lemma. If $n \geq 2$ then pumping lemma is trivially true.

7.2.3 Let 'L' denote that language and suppose that 'L' is context free. Let 'p' be the pumping length. Consider the string $w = 1^p \# 1^p \# 1^{2p}$. Clearly, $w \in L$ and $|w| \geq p$. Therefore, according to the pumping lemma, w can be written as uvxyz where

* v y is not equal to epsilon.
* $uv^k xy^k z \in L$, for every $k \geq 0$.

Now, there are several cases to consider.

First, suppose that either v or y contains a #. Then $uv^2xy^2z$ is not equal to L, because it contains too many #'s.

For the remaining cases assume that neither v, nor y contains a #. Note that 'w' consists of 3 blocks of 1's seperated by #'s. This implies that v and y one each completely contained within one block and that v and y cannot contain 1's from all 3 block

The second case is when v and y don't contain any 1's from the third block. Then $uv^2xy^2z = 1^{p+i} \# 1^{p+j} \# 1^{2p}$, where atleast one of i, j is greater than 0. This implies that $uv^2xy^2z$ not equal to L.

The third case is when v and y don't contain any 1's from the first two blocks. Then $uv^2xy^2z = 1^p \# 1^p \# 1^{2p+i}$ where $i > 0$. This implies that $uv^2xy^2z$ not equal to L.

The fourth case is when v consits of 1's the first block and y consists of 1's the third block. Then $uv^2xy^2z = 1^{p+i} \# 1^p \# 1^{2p+j}$ where both i, j are greater than 0. This implies that $uv^2xy^2z$ is not equal to L because the first block is greater than the second block.

The fifth and final case is when v consists of 1's the second block and y consists of 1's the third block. Then $uv^0xy^0z = 1^p \# 1^{p-i} \# 1^{2p-j}$, where both i, j are greater than 0. This implies that $uv^2xy^2z$ is not equal to L because the second block is smaller than the first block. In all cases, we have a contradiction.

This proves that 'L' is not context free

7.2.5 a) We begin selecting $z = 0^{2n} 1^{2n} 0^{\wedge}$ & marking all the last 0's. If we select v or x to have both 0's and 1's in it, we can instantly see that one syntax is no longer correct. If we select 'v' to be 0/1 there we can also see that one original assumption that $j = \max(i, k)$ no longer holds because one of the ~~as~~ numbers will grow while the other will remain the same. One only choice is to have 'v' & 'x' exclusively containing $\hat{0}$. when we start pumping it, at some point the number of $\hat{0}$ (corresponding to k) will grow to be larger than i and j, therefore once again breaking one condition that $j = \max(i, k)$

∴ The language is not a CFL//.

7.2.5 b) Consider the case where we choose $z = a^x b^x c^{x!+x}$. We mark all the a's & b's. We first note that if 'v' or 'x' contain a mix of a's & b's then we can see that with $i = 2$,

the structure of the resulting grammar is no longer correct. We now look at the case where $V = a^{\alpha}$ & $x = b^{\beta}$ ij $\alpha \neq \beta$ and see that one final string is of the form

$$a^{x+\gamma(i-1)} b^{n+\gamma(i-1)} c^{n!+n}$$

If we set the exponents of a & b equal to 'c', let $N_{ijA}$ denote the number of distinct parse trees for substring $a_1 \ldots a_j$ of input 'w'. Starting from variable A (A ⇒ root of the tree). Note that 'A' would be a metavariable & not a variable in G. $N_{1ns}$, where $n = |w|$ and S, the starting variable of $G_1$ is the value we are interested in. We can argument that CYK algorithm to compute each $N_{ijA}$ as we compute the corresponding $X_{ij}$. Then we proceed to compute $N_{ijA}$ for each variable A.

Initially we set all $N_{ijA}$ to 0.

$$N + \gamma(i-1) = n! + x$$

$$\gamma(i-1) = n!$$

$$i - 1 = \frac{n!}{\gamma}$$

Since $\delta \leq n$, wkt the R.H.S devides evenly and therefore we can pick any $i$ that satisfies this constraint, therefore one original constraint of ~~$\{a^n b^n c^n | i \neq n\}$~~ $\{a^n b^n c^n | i \neq n\}$ is not satisfied, therefore we do not have a CFL. //

T.4.1 b) Apply (a) first, and assume without loss of generality G is already in CNF. If $L(G)$ is infinite then return 'YES'. Else enumerate all possible derivations from 'S'. If there are atleast 100 different strings return 'YES' else 'NO'.

T.4.2 a) Fix a symbol ~~$a \in \Sigma$~~ $a \in \Sigma$. We want to know whether or not 'a' appears in some sential form $\gamma \in (V \cup T)^*$. It suffices to know whether or not 'a' appears in some sentential form $\delta \in (V \cup T)^*$. It suffices to know whether or not 'a' appears in each variable. Construct a graph with vertices $V \cup \{a\}$.

If there is a production rule $A \to \gamma$, where $\delta$ contains variable 'B' then add an edge $A \to B$. It is easy to check that 'a' can appear in variable A if & only if there is a path from A to a. Both the construction &

reachability test can be done in linear time.

7.4.2 b) Initially mark all variables 'A' with the production $A \to \varepsilon$. If there's a production $A \to B_1 B_2 \ldots B_n$, where all $B_i$'s are nullable then mount 'A' as nullable. This can be implemented in linear time.

7.4.3 b)

| | | | | |
|---|---|---|---|---|
| {S, C} | | | | |
| {S, A, C} | {S, C} | | | |
| $\phi$ | {S, A, C} | {B} | | |
| {S, A} | {B} | {B} | {S, C} | |
| {B} | {A, C} | {A, C} | {A, C} | {B} |
| b | a | a | a | b |

'S' belongs to the uppermost set, which means that the word is generated by the grammar since 'S' is the start symbol of grammar.

7.4.3 c) Using CYK, to determine whether 'aabab' is in the language of the grammar 'G':

$S \to AB \mid BC$

$A \to BA \mid a$

$B \to CC \mid b$

$C \to AB \mid b$

## Matrix construction:

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | | A, C | B | B | S, A, C | S, C |
| 1 | | | A, C | S, C | B | B |
| 2 | | | | B | S, A | S, C |
| 3 | | | | | A, C | S, C |
| 4 | | | | | | B |
| 5 | | | | | | |

| a | a | b | a | b |
|---|---|---|---|---|

Since, cell at the upper right hand corner (0,5) contains the start symbol, the string 'aabab' is in L(G).

T.4.5 >

Associate an integer with every element of every $X_{ij}$ that appears in the CYK algorithm's table. Modify the CYK algorithm's basis to set all of the integers in the first row to 1. Modify the induction step to include the following:

For every production of the form $P \rightarrow QR$ found by the CYK algorithm for $X_{ij}$, let P be the current integer associated in 'P' with P in $X_{ij}$. let Q be the integer associated with Q in $X_{ik}$, and let r be the integer associated with 'R' in $X_{k+1,j}$ then update 'P' to be P+Q.r.

If the start symbol (S) is not in $X_{1n}$ where 'n' is the length of the input string then return 0. Otherwise return the integer associated with start symbol in $X_{1n}$.

7.3.1 c) For each variable A of the original grammar G, let A' be a new variable that generates initially of what 'A' generates. Thus, if 'S' is the start symbol of G, we make S' the new start symbol. If $A \to BC$ is a production of G, then in the new grammar we have $A \to BC$, $A' \to BC'$ and $A' \to B'$. If $A \to a$ is a production of G, then the new grammar has $A \to a$, $A' \to a$ and $A' \to \epsilon$.

It can be seen that CFL are closed under the cycle.

7.3.2 Let 'L' denote the language & suppose that 'L' is context-free. Let 'p' be the pumping length. Consider the string $w = 1^p \# 1^p \# 1^{p^2}$. Clearly $w \in L$ and $|w| > p$. Therefore according to the pumping lemma 'w' can be written as uvxyz where

i. $vy \neq \epsilon$

ii. $uv^k xy^k z \in L$, for every $k \geq 0$

There are several cases to consider.

Case 1: v & y are both within the third block. Then $uv^2xy^2z = 1^p \# 1^p \# 1^{p^2+i}$ where $i > 0$. This implies that $uv^2xy^2z \notin L$

Case 2: where 'v' consists of 1's from the first block & 'y' consists of 1 from the third block. This case cannot occur since $|vxy| \leq p$.

In all the cases we have contradictio-n

Hence, 'L' is not context-free.

7.3.3 b) Max:

Consider $L = \{a^i b^j c^k \mid i \leq j \text{ or } i \leq k\}$ to be a CFL

Then $\max(L) = \{a^i b^j c^k \mid k = \max(i,j)\}$

Use the pumping lemma with the string $a^n b^n c^n$ to show this is not a CFL. Hence CFL's are not closed under max.

7.3.3 c) Half:

Take $L = \{a^n b^n c^m d^{3m} \mid m, n \geq 0\}$ to be a CFL.

Now, take $half(L) \cap a^* b^* c^+$ & this will give you only cases where $n < m$

$L' = \{a^n b^n c^j \mid j < n\}$

Clearly this is not a CFL & hence

Half is not a closure property.

7.3.3 d) Alt:

Let us start with the string of the form $0^n 1^n$ & put any number of 0's. We can obtain any strings of 0's & 1's that begin with atleast as many 0's as there are 1's in the entire string.

7.3.4 a) Shuffle (00, 111)

{00111, 01110, 11100, 10011, 11001, 01011, 01101, 10101, 10110}

7.3.4 d) Proof by machine construction:

Since R is a regular language, it has a DFA Let us construct a CFL to accept shuffle (L, R) as follows.

$Q = Q_L \times Q_R$, $\Sigma = \Sigma$, $\Gamma = \Gamma$, $q_0 = q_{0L} \times q_{0R}$, $Z_0 = Z_{0L}$

$F = F_L \times F_R$

The transition function is easier to describe. On each read of the input tape, the newly constructed machine guesses whether the input came from L or R. and the transitions accordingly.

Since we can construct the PDA for shuffle ~~we can show that~~ it holds that it is a

Context free language

T-3.4 e)

Take $L_1 = \{a^n b^n \mid n>0\}$ & $L_2 = \{c^m d^m \mid m>0\}$

Now shuffle $(L_1 \& L_2) \cap a^* b^* c^* d^*$ will give us all strings of the form $\{a^n c^m b^n d^m \mid m, n>0\}$ which is clearly not a CFL. Since we know that the CFL's are closed under union with a regular set and one set is clearly regular, it holds that CFL's are not closed under shuffle.

T-3.6 If 'L' is a CFL with grammar 'G', from a grammar for $L^R$ by reversing the right side of every production.

Example: Let 'G' have $S \to 0S1 \mid 01$

The reversal of L(G) has grammar $S \to 1S0 \mid 10$.