

# Chomsky Normal Form

The Pumping Lemma for Regular Languages came from properties of the automaton associated with a regular language. The corresponding lemma for context-free languages comes from properties of the associated grammar. To get at those properties we need to write the grammar in a specific way.

A grammar is said to be in Chomsky Normal Form (CNF) if all of its grammar rules follow one of the two patterns:

- $X \Rightarrow YZ$  (exactly 2 non-terminals on the right side)
- $X \Rightarrow a$  (exactly 1 terminal on the right side)

We will show that every context-free grammar can be converted into a CNF grammar that defines the same language.

Step 1. We say nonterminal  $X$  is *generating* if there is a terminal string  $w$  with  $X \xRightarrow{*} w$ . We say  $X$  is *reachable* if there is a derivation from the start symbol that contains  $X$ :

$$S \xRightarrow{*} \alpha X \beta \quad \text{where } \alpha, \beta \text{ are in } (\Sigma + N)^*$$

Note that if a symbol is not generating or not reachable then we can remove it from the grammar along with all rules that contain it.

Algorithm: To mark all generating variables

- a) Mark all symbols that have a rule where the right hand side contains only terminal symbols.
- b) Mark all symbols  $X$  for which there is a rule  $X \Rightarrow \alpha$  where every symbol in  $\alpha$  is either marked or a terminal symbol.

repeat step b until nothing more can be marked.

An easy induction shows that symbol  $X$  is marked if and only if it is generating.

For example, consider the grammar

$$S \Rightarrow AB \mid a$$
$$A \Rightarrow C$$
$$C \Rightarrow b$$
$$B \Rightarrow Eb$$

Symbols  $S$ ,  $A$ , and  $C$  are marked;  $B$  and  $E$  are not.

Algorithm: To mark the reachable symbols:

- a) Mark the start symbol  $S$
- b) If  $X$  is marked then for each rule  $X \Rightarrow \alpha$  mark all of the nonterminal symbols in  $\alpha$ .

Again, an easy induction shows that a symbol is marked if and only if it is reachable.

Example:

$S \Rightarrow AB \mid a$

$A \Rightarrow BC \mid a$

$B \Rightarrow b$

$C \Rightarrow BA$

$D \Rightarrow b$

S,A,B,C are all reachable; D is not.



The order in which we remove rules and symbols from a grammar matters.

Example:

$$S \Rightarrow AB \mid a$$
$$A \Rightarrow b$$

Here all variables are reachable but B is not generating.

If we remove unreachable variables then non-generating ones we get

$$S \Rightarrow a$$
$$A \Rightarrow b$$

If we remove non-generating variables then unreachable ones we get

$$S \Rightarrow a$$

Theorem: Let  $G$  be a context-free grammar that derives a non-empty language.

Step 1: First eliminate all symbols that aren't generating and all rules that use them. Call this grammar  $G'$

Step 2: Then eliminate all symbols that aren't reachable in the grammar produced by Step 1, and all rules that use them. Call this grammar  $G''$ .

The language derived from  $G''$  is the same as the language derived from  $G$  and all of the non-terminal symbols in  $G''$  are both reachable and generating.

Proof: One direction is easy:  $G''$  is a subset of  $G$ , so everything that can be derived from  $G''$  can also be derived from  $G$ .

So suppose  $w$  can be derived from  $G$ . This means there is a derivation  $S \xRightarrow{*} w$ . Every variable used in this derivation is reachable (since it is derived from  $S$ ) and generating (since it derives a terminal string). So  $w$  can also be derived from  $S$  in  $G'$ . This shows the languages of  $G$  and  $G'$  are the same.

Since we do Step 2 last it is obvious that all of the symbols in  $G''$  are reachable. We need to show that they are still generating in  $G''$ .

Suppose  $X$  is a symbol in  $G''$ . If  $X$  wasn't removed in Step 2 then  $X$  must be reachable in  $G''$   $S \xRightarrow{*} \alpha X \beta$  for some strings  $\alpha$  and  $\beta$  of symbols in  $G''$ .

All of the symbols in  $\alpha X \beta$  weren't removed in Step 1 so they must be generating in  $G$ : there are strings  $w_1$ ,  $w_2$  and  $w_3$  of terminal symbols where  $\alpha \xRightarrow{*} w_1$ ,  $X \xRightarrow{*} w_2$ , and  $\beta \xRightarrow{*} w_3$ . So now we have  $S \xRightarrow{*} \alpha X \beta \xRightarrow{*} w_1 w_2 w_3$ , and all of the symbols used in the derivation  $X \xRightarrow{*} w_2$  are both reachable and generating so they remain in  $G''$ . This means that  $X$  is generating in  $G''$ .

Chomsky Normal Form doesn't allow rules  $A \Rightarrow \varepsilon$ .

Definition: We say symbol  $A$  is *nullable* if  $A \overset{*}{\Rightarrow} \varepsilon$

Here is a marking algorithm to mark the nullable symbols:

- a) Mark  $A$  if there is a rule  $A \Rightarrow \varepsilon$ .
- b) Mark  $B$  if there is a rule  $B \Rightarrow A_1..A_k$  and all of the symbols on the right hand side are marked.

Repeat (b) until nothing else can be marked.

It is easy to see that this does mark the nullable symbols and only the nullable symbols.

Theorem: Let  $G$  be a grammar.

1. Eliminate all rules of the form  $A \Rightarrow \varepsilon$
2. If there is a rule  $X \Rightarrow \alpha A \beta$  where  $A$  is the only nullable symbol on the right hand side, then replace this rule by
$$X \Rightarrow \alpha A \beta \mid \alpha \beta$$
3. If a rule has  $m$  nullable variables on its right hand side, replace it with  $2^m$  rules having the nullable variables present or absent in all possible combinations.

Let  $G'$  be the grammar this produces. Then  $G'$  has no nullable symbols and generates the same language as  $G$  except for the empty string. Note that  $G'$  might have variables that are no longer generating.

Since we have eliminated all rules  $A \Rightarrow \varepsilon$  there is no way for  $G'$  to derive  $\varepsilon$ , so  $G'$  has no nullable symbols.

If  $G'$  derives string  $w$ , then any step in the derivation using a rule modified in (2) or (3) could be replaced by the original rule, producing a derivation of  $w$  in  $G$ . So any string that can be derived in  $G'$  can be derived in  $G$ .

We will show by induction that any string  $w$  other than  $\varepsilon$  that can be derived in  $G$  can be derived in  $G'$ .

The induction is on the length of the derivation in  $G$ . If this is 1 the derivation must be  $A \Rightarrow w$ , which does not use anything modified in  $G'$ .

Suppose this is true for all derivations in  $G$  of length  $\leq n$  steps and we have a derivation of  $w$  taking  $n+1$  steps. The first step of this must be of the form  $A \Rightarrow X_1..X_k$ . Since this derivation eventually produces  $w$ , each  $X_i$  must derive a string  $w_i$  in  $n$  or fewer steps. If  $w_i = \varepsilon$  then  $X_i$  is nullable and there is an equivalent rule in  $G'$  without  $X_i$ . If  $w_i$  is not empty the inductive hypothesis says  $X_i$  can derive  $w_i$  in  $G'$ . Either way,  $w$  can be derived from  $A$  in  $G'$ .



Chomsky Normal Form doesn't allow rules of the form  $A \Rightarrow B$ , where  $B$  is a single symbol. We call such a rule a *unit production*.

If  $A \xRightarrow{*} C$  using only unit productions (as in  $A \Rightarrow B$  and  $B \Rightarrow C$ ) we call  $(A, C)$  a *unit pair*.

Here is an algorithm to mark the unit pairs of a grammar:

Algorithm:

- 1) Mark  $(A, A)$  for every nonterminal symbol  $A$ .
- 2) If  $(A, B)$  is marked and  $B \Rightarrow C$  is a unit production then mark  $(A, C)$

Repeat (2) until nothing else can be marked.

Here is an algorithm for removing the unit productions from a grammar  $G$ , producing a new grammar  $G'$

- 1) Start  $G'$  with no grammar rules.
- 2) For each unit pair  $(A,B)$  in  $G$  and each non-unit rule  $B \Rightarrow \alpha$  in  $G$ , add  $A \Rightarrow \alpha$  as a rule to  $G'$ .

Note that since we defined  $(A,A)$  as a unit pair, rule (2) adds all non-unit rules of  $G$  to  $G'$

Example:

$A \Rightarrow B$

$B \Rightarrow 0C \mid 1D$

$C \Rightarrow 0$

$D \Rightarrow 1$

This is equivalent to

$A \Rightarrow 0C \mid 1D$

$C \Rightarrow 0$

$D \Rightarrow 1$

We eliminate B because it is no longer reachable.

Theorem: If  $G'$  is the grammar produced from  $G$  by this algorithm that removes unit pairs then  $G'$  and  $G$  derive the same language.

Proof: It is obvious that every derivation in  $G'$  is a shortcut for one in  $G$ , so every string derived in  $G'$  can be derived in  $G$ . Now suppose that  $w$  is derived in  $G$ . Consider the left-most derivation of  $w$ . If at some step this uses a unit production  $A \Rightarrow B$  then at the next step  $B$  will be the leftmost nonterminal symbol so it will be expanded with a rule  $B \Rightarrow \alpha$ . We could get to this same point in  $G'$  by using the rule  $A \Rightarrow \alpha$ . So every string derived in  $G$  can also be derived in  $G'$ .

Recall that CNF allows only rules of the form  $A \Rightarrow BC$  or  $A \Rightarrow a$

Algorithm: To convert a grammar into Chomsky Normal Form:

- 1) Eliminate any  $\varepsilon$ -rules
- 2) Eliminate any unit rules
- 3) Eliminate any rules that are not generating
- 4) Eliminate any rules that are not reachable
- 5) For each rule  $A \Rightarrow X_1 \dots X_n$  where  $n > 1$ , if some  $X_i$  is a terminal symbol then add a new nonterminal symbol  $A_i$  to the grammar and the rule  $A_i \Rightarrow X_i$ . Replace the original rule with  $A \Rightarrow X_1 \dots X_{i-1} A_i X_{i+1} \dots X_n$ . So we can assume the  $X_i$  are all nonterminals
- 6) (over)

6) For each rule  $A \Rightarrow X_1..X_n$  where  $n > 2$  make a new set of rules

$$A \Rightarrow X_1 B_1$$

$$B_1 \Rightarrow X_2 B_2$$

...

$$B_{n-3} \Rightarrow X_{n-2} B_{n-2}$$

$$B_{n-2} \Rightarrow X_{n-1} X_{n-2}$$

where the  $B_i$  are new nonterminal symbols

Call the new grammar  $G'$ . It should be obvious from everything we have done that  $\mathcal{L}(G') = \mathcal{L}(G) - \{\varepsilon\}$  and  $G'$  is in Chomsky Normal Form.

So every context-free language has a CNF grammar that derives all of the language except  $\{\varepsilon\}$

Example:

$$E \Rightarrow E+T \mid T$$

$$T \Rightarrow T^*F \mid F$$

$$F \Rightarrow (E) \mid \text{digit} \mid F \text{ digit}$$

---

Step 1: No  $\varepsilon$ -rules

Step 2: Unit rules

$$E \Rightarrow E+T \mid T^*F \mid (E) \mid \text{digit} \mid F \text{ digit}$$

$$T \Rightarrow T^*F \mid (E) \mid \text{digit} \mid F \text{ digit}$$

$$F \Rightarrow (E) \mid \text{digit} \mid F \text{ digit}$$

Steps 3, 4: All symbols are reachable and generating

Step 5:

$E \Rightarrow EPT \mid TXF \mid LER \mid \text{digit} \mid FD$

$T \Rightarrow TXF \mid LER \mid \text{digit} \mid FD$

$F \Rightarrow LER \mid \text{digit} \mid FD$

$P \Rightarrow +$

$X \Rightarrow *$

$L \Rightarrow ($

$R \Rightarrow )$

$D \Rightarrow \text{digit}$

Step 6: over



Step 6:

$$E \Rightarrow E E_1 \mid T T_1 \mid L L_1 \mid \text{digit} \mid FD$$
$$E_1 \Rightarrow P T$$
$$T_1 \Rightarrow X F$$
$$L_1 \Rightarrow E R$$
$$T \Rightarrow T T_1 \mid L L_1 \mid \text{digit} \mid FD$$
$$F \Rightarrow L L_1 \mid \text{digit} \mid FD$$
$$P \Rightarrow +$$
$$X \Rightarrow *$$
$$L \Rightarrow ($$
$$R \Rightarrow )$$
$$D \Rightarrow \text{digit}$$

This grammar is in CNF.