CSCI 341–Fall 2024: Lecture Notes
Set 9: Chomsky Normal Form

Edward Talmage

October 7, 2024

A simplified, restricted format for CFGs can make it easier to treat them formally or automatically. For example, writing an algorithm or construction to do something with a CFG is easier when the grammar is more predictable, similar to how we tend to use DFAs in constructions, though we often build NFAs out of them. One such simplified format for CFGs is *Chomsky Normal Form*.

Definition 1. A Context-Free Grammar is in *Chomsky Normal Form* if

1. Every rule is of the form $A \to BC$ or $A \to a$, where $A, B, C \in V, a \in \Sigma$.
2. S is not on the right hand side of any rule.
3. $S \to \varepsilon$ is also allowed (exception to rule 1).

Of course, this form is only of use to us if it is fully expressive. That is, we want to know that for every CFL, there is a CFG in Chomsky Normal Form which generates that language.

Theorem 1. *Every CFG can be converted to an CNF.*

Idea: In order, add new start variable to ensure it is not generated by any rule, remove ε-rules, remove singleton rules, split apart combination rules.

Running Example: Consider the grammar

$$\begin{aligned} S &\to A \mid B \\ A &\to aBB \mid \varepsilon \\ B &\to \varepsilon \mid b \end{aligned}$$

Proof. To prove the theorem by construction, we give an algorithm for converting any CFG to an equivalent grammar in CNF.

1. Add a new start variable S', with rule $S' \to S$. This satisfies rule 2.
2. Remove all rules of the form $A \to \varepsilon$, for $A \neq S'$.
   - To avoid changing the grammar's behavior, replace all rules generating A, $B \to CADAE$, in a way that handles the possibility of A generating the empty string:
   $$B \to CADAE \mid CDAE \mid CADE \mid CDE$$
   - Rules $B \to A$ become $B \to \varepsilon$, unless we have already removed $B \to \varepsilon$.

   Exercise: Why do we not need to re-handle the case where $B \Rightarrow A \Rightarrow \varepsilon$?

   - This process terminates when there are no more ε rules, or the only one left is $S' \to \varepsilon$, which is allowed by condition 3
3. Remove all rules $A \to B$. Replace with $A \to w$, for every $B \to w$, unless we have already deleted $A \to w$.
4. Replace rules $A \to u_1u_2 \cdots u_k, k \geq 2$ with $A \to U_1A_1, A_1 \to U2A2, \ldots, A_{k-2} \to U_{k-1}U_k$, where

- Each A_i is a new variable.
- If u_i is a variable, $U_i = u_i$. (This handles rules $A \to BC$, since we leave the two variables alone.)
- If u_i is a terminal, U_i is a new variable, and we add $U_i \to u_i$.

This last step is replacing a chain of variables and/or terminals by a sequence of expansions, each splitting off the first element of the chain. Thus, if A yields a chain of n elements, U_1 yields the first element, then A_1 generates all but the first element of the chain, by way of U_2 yielding the second element and A_2 generating all but the first two elements of the chain, and so on.

Since at every step we have maintained the set of strings the grammar can generate, and have left only rules allowed in CNF, we have constructed a CFG in CNF equivalent to our original grammar. □

Running Example, continued: After the first two steps, we will have the following grammar. Note that we added and subsequently remove $S \to \varepsilon$, since either A or B could generate the empty string, but S is no longer the start variable, so we cannot leave it yielding ε.

$$\begin{aligned} S' &\to S \mid \varepsilon \\ S &\to A \mid B \\ A &\to aBB \mid aB \mid a \\ B &\to b \end{aligned}$$

Next, we remove all singleton rules, in which one variable directly generates another, and get

$$\begin{aligned} S' &\to \varepsilon \mid aBB \mid aB \mid a \mid b \\ S &\to aBB \mid aB \mid a \mid b \\ A &\to aBB \mid aB \mid a \\ B &\to b \end{aligned}$$

At this point, there starts to be a lot of redundant rules, since S' can generate anything S can, and S can generate anything A or B can. To keep this legible, we will reuse U_1 and A_1 that step 4 generates for each of the different appearances of the same right hand side of a rule, and get the following CNF grammar equivalent to the input grammar:

$$\begin{aligned} S' &\to \varepsilon \mid U_1A_1 \mid U_1B \mid a \mid b \\ S &\to U_1A_1 \mid U_1B \mid a \mid b \\ A &\to U_1A_1 \mid U_1B \mid a \\ B &\to b \\ U_1 &\to a \\ A_1 &\to BB \end{aligned}$$

As with many of our constructions in this class, this is not an "efficient" construction, in that it has many redundant derivations, may have unreachable variables, and so on. But we are not concerned with performance, merely with the fact that there is always a grammar in CNF equivalent to any CFG. Thus, if we need to construct something, we may assume that a CFL has a CFG in CNF, which can make our constructions far easier.

1 Practice

Exercise: Convert the following CFG to CNF:

$$\begin{aligned} A &\to BAB \mid B \\ B &\to 00 \mid \varepsilon \end{aligned}$$

1. Add $S' \to A$

2. Remove ε-rules:
   (a) Remove $B \to \varepsilon$. Add $A \to \varepsilon \mid AB \mid BA \mid A$.
   (b) Remove $A \to \varepsilon$. Add $S' \to \varepsilon, A \to BB$. (Don't miss the place where A can yield itself.)
3. Remove singleton rules:
   (a) Remove $A \to A$. Add nothing.
   (b) Remove $A \to B$. Add $A \to 00$.
   (c) Remove $S \to A$. Add $S \to BAB \mid AB \mid BA \mid BB \mid 00$.

At this point, the grammar looks like this:

$$\begin{aligned} S &\to \varepsilon \mid BAB \mid AB \mid BA \mid BB \mid 00 \\ A &\to BAB \mid AB \mid BA \mid BB \mid 00 \\ B &\to 00 \end{aligned}$$

4. Remove compound rules:
   (a) Remove $B \to 00$, add $B \to ZZ, Z \to 0$.
   (b) Remove $A \to BAB$, add $A \to BA', A' \to AB$.
   (c) Remove $A \to 00$, add $A \to ZZ$.
   (d) Remove $S \to BAB$, add $S \to BA'$.
   (e) Remove $S \to 00$, add $S \to ZZ$.

Result:

$$\begin{aligned} S &\to \varepsilon \mid BA' \mid AB \mid BA \mid BB \mid ZZ \\ A &\to BA' \mid AB \mid BA \mid BB \mid ZZ \\ A' &\to AB \\ B &\to ZZ \\ Z &\to 0 \end{aligned}$$

Exercise: Convert the following CFG to CNF:

$$\begin{aligned} S &\to ASA \mid aB \\ A &\to B \mid S \\ B &\to b \mid \varepsilon \end{aligned}$$

1. Add new start variable: $S' \to S$
2. Remove ε-rules:
   (a) Remove $B \to \varepsilon$, add $A \to \varepsilon, S \to a$.
   (b) Remove $A \to \varepsilon$, add $S \to SA \mid AS \mid S$.

   Grammar is currently

$$\begin{aligned} S' &\to S \\ S &\to ASA \mid SA \mid AS \mid S \mid aB \mid a \\ A &\to B \mid S \\ B &\to b \end{aligned}$$

3. Remove singleton rules:

(a) Remove $A \to B$, add $A \to b$.

(b) Remove $A \to S$, add $A \to ASA \mid SA \mid AS \mid aB \mid a$ (Note that we do not add $A \to S$, as we have removed that already.)

(c) Remove $S \to S$, add none.

(d) Remove $S' \to S$, add $S' \to ASAS \mid SA \mid AS \mid aB \mid a$.

Grammar is now

$$\begin{aligned}
S' &\to ASA \mid SA \mid AS \mid aB \mid a \\
S &\to ASA \mid SA \mid AS \mid aB \mid a \\
A &\to ASA \mid SA \mid AS \mid aB \mid a \mid \mid b \\
B &\to b
\end{aligned}$$

4. Remove compound rules:

(a) Remove $A \to ASA$, add $A \to AT$, $T \to SA$.

(b) Remove $A \to aB$, add $A \to A'B$, $A' \to a$.

(c) Remove $S \to ASA$, add $S \to AT$.

(d) Remove $S \to aB$, add $S \to A'B$.

(e) Remove $S' \to ASA$, add $S' \to AT$.

(f) Remove $S' \to aB$, add $S' \to A'B$.

This yields the final grammar in Chomsky Normal Form:

$$\begin{aligned}
S' &\to AT \mid SA \mid AS \mid A'B \mid a \\
S &\to AT \mid SA \mid AS \mid A'B \mid a \\
A &\to AT \mid SA \mid AS \mid A'B \mid a \mid b \\
B &\to b \\
A' &\to a \\
T &\to SA
\end{aligned}$$