Consider the grammar:

\[
\begin{align*}
A & \rightarrow B \mid a \mid CBD \\
B & \rightarrow C \mid b \\
C & \rightarrow A \mid c \\
D & \rightarrow d
\end{align*}
\]

Some strings in the language of this grammar are: \(a, \quad cbd, \) etc. Notice that this grammar is not immediately left-recursive in that there is no single production \(X \rightarrow Xa\). However, it is left-recursive because there are valid derivations of the form \(A \rightarrow^* Aa\) (and \(B \rightarrow^* B\beta\) and \(C \rightarrow^* C\delta\)). Let’s demonstrate one: \(A \rightarrow B \rightarrow C \rightarrow A\), so \(A \rightarrow^* A\).

To warm up, let’s compute \(\text{First()}\) and \(\text{Follow()}\) sets for this grammar.

\(\text{First}(A)\) must contain \(\text{First}(B), \{a\}\) and \(\text{First}(CBD)\). \(\text{First}(B)\) must contain \(\text{First}(C)\) and \(\{b\}\). \(\text{First}(C)\) must contain \(\text{First}(A)\) and \(\{c\}\). \(\text{First}(CBD) = \text{First}(C)\) as before. So \(\text{First}(A) = \{a, b, c\}\). Similarly, \(\text{First}(B) = \text{First}(C) \cup \{b\} = \{a, b, c, d\}\) and \(\text{First}(C) = \text{First}(A) \cup \{c\} = \{a, b, c\}\). \(\text{First}(D) = \{d\}\).

To compute \(\text{Follow}(A)\) we look for every occurrence of \(A\) on the right-hand side of a production. We find one in \(C \rightarrow A\), but it that \(A\) is at the direct end of the production, we get that \(\text{Follow}(A)\) includes \(\text{Follow}(C)\). Now we look for \(C\) on the right-hand side of a production and find it in \(A \rightarrow CBD\). This time there is something to the right of \(C\), so we get that \(\text{Follow}(C)\) contains \(\text{First}(B)\). However, we also see a \(C\) on the right of \(B \rightarrow C\), so \(\text{Follow}(C)\) contains \(\text{Follow}(B)\). Looking for \(B\)’s on the right we find \(A \rightarrow CBD\), so \(\text{Follow}(B)\) contains \(\text{First}(D) = \{d\}\). So \(\text{Follow}(A) = \text{Follow}(B) = \text{Follow}(C) = \{a, b, c, d\}\). Since \(D\) appears in the production \(A \rightarrow CBD\), we have that \(\text{Follow}(D)\) includes \(\text{Follow}(A)\), so \(\text{Follow}(D) = \{a, b, c, d\}\) as well.

OK, messy grammar. Now let’s eliminate left-recursion. The first step is to make all left-recursion immediate by doing some substitutions. For example, since we have \(A \rightarrow B \rightarrow C \rightarrow A\), we need to take the production \(A \rightarrow B\) and replace it with \(A \rightarrow C\) and \(A \rightarrow B\). That gives us:

\[
A \rightarrow C \mid b \mid a \mid CBD
\]

But we’re not done, since we can have \(A \rightarrow C \rightarrow A\). So now we need to substitute in for \(C\) in that production. Let’s do that once:

\[
A \rightarrow A \mid c \mid b \mid a \mid CBD
\]

To get here, we just removed the production \(A \rightarrow C\) and replaced it by \(A \rightarrow A\) and \(A \rightarrow c\) (we got those two right-hand sides from the productions \(C \rightarrow A\) and \(C \rightarrow c\)). Now we’re almost done, just one more possible non-immediate left-recursion. Let’s substitute it away:

\[
A \rightarrow A \mid c \mid b \mid a \mid ABD \mid cBD
\]

Huzzah! Now \(A\) has only immediate left-recursion. And actually, there is no other left-recursion left in the grammar now, since we can no longer derive \(B \rightarrow^* B\beta\) or \(C \rightarrow^* C\delta\). So we can leave the \(B\) and \(C\) and \(D\) productions alone and concentrate on eliminating left-recursion from \(A\).

First, let’s group the productions into those that are left-recursive and those that are not:

\[
A \rightarrow A \mid ABD
\mid c \mid b \mid a \mid cBD
\]

Now imagine that you actually have this grammar before you. You can expand things for a long time by just using the first to productions: \(A \rightarrow ABD \rightarrow ABDBD \rightarrow ABDBDBD\), etc. But eventually you have to settle down and use
one of the other productions: $A \rightarrow ABD \rightarrow ABDBD \rightarrow cBDBD$ and then the chain stops. So we get the idea that $A$ can eventually produce something that starts with $c, b, a$ or $cBD$ and ends with a list of $BD$’s.

One other thing to note is that the production $A \rightarrow A$ itself is useless — it does not change the language of the grammar and can be safely dropped. So here’s the revised left-recursive grammar:

$$A \rightarrow ABD$$
$$\quad | \quad c | b | a | cBD$$

Now let’s break that down into $A$ and $A’$. We reasoned above that an $A$ goes to $c|b|a|cBD$ followed by a list of $BD$’s. Let’s make the first bit the $A$ and make the list of $BD$’s the $A’$.

$$A \rightarrow cA’ | bA’ | aA’ | cBD\ A’$$

$$A’ \rightarrow \varepsilon | BDA$$

We’re done. You can check and see that every production for $A$ is of the form $A \rightarrow A’$ and that $A’$ really does define a (possibly empty) list of $BD$’s. Let’s do one more example, just for fun. Consider the grammar:

$$Q \rightarrow QED | q$$
$$E \rightarrow e$$
$$D \rightarrow NFA | d$$
$$N \rightarrow DFA | n$$
$$F \rightarrow f$$
$$A \rightarrow a$$

This grammar is left recursive. In fact, it is immediately left-recursive in one place and non-immediately left-recursive in two places. First, let’s substitute to get rid of the non-immediate left recursion. Consider the derivation: $D \rightarrow NFA \rightarrow DFAFA$. Since it ends up being left recursive, we must substitute. Take the production $D \rightarrow NFA$ and remove it. Then for every production $N \rightarrow \alpha_i$, add a production $D \rightarrow \alpha_iFA$. That gives us:

$$D \rightarrow DFAFA | nFA | d$$

Notice again that in one fell swoop we have eliminated the whole chain of non-immediate left-recursion: we can no longer derive $N \rightarrow^* N\beta$. So now our grammar looks like:

$$Q \rightarrow QED | q$$
$$E \rightarrow e$$
$$D \rightarrow DFAFA | nFA | d$$
$$N \rightarrow DFA | n$$
$$F \rightarrow f$$
$$A \rightarrow a$$

Now it’s time to eliminate the immediate left recursion. Let’s start with $Q \rightarrow QED | q$. Once again, we can derive strings like $Q \rightarrow QEDQED$, but eventually we have to stop and use $Q \rightarrow q$. Taking $Q$ to be the $q$ bit and $Q’$ to be the list of $ED$’s, we get:

$$Q \rightarrow qQ’$$
$$Q’ \rightarrow \varepsilon | EDQ’$$
Now let’s look at $D \to DFAD\epsilon|nFA|d$. We see that we can make a huge list of $FAFA'$'s using the first production but we eventually have to start with $nFA$ or $d$. Let’s make $D'$ the list and $D$ the first bit.

\[
\begin{align*}
  D & \to nFAD' | dD' \\
  D' & \to \epsilon | FAFAD'
\end{align*}
\]

OK, that’s all the left-recursion. The final grammar is:

\[
\begin{align*}
  Q & \to qQ' \\
  Q' & \to \epsilon | EDQ' \\
  E & \to e \\
  D & \to nFAD' | dD' \\
  D' & \to \epsilon | FAFAD' \\
  N & \to DFA | n \\
  F & \to f \\
  A & \to a
\end{align*}
\]

Huzzah, we are done.