Context Free Grammar Exercises:

Exercise 1:

Consider the CFG G with non-terminals {E,T }, terminals {a, b,c,d,e}, start symbol E and productions:

E → Eabc
E→Ecde
E → T
T→bc

Transform G into an equivalent CFG G′ without left recursion.

Solution:

Remember the rule we have seen in class:

<foo> →<foo>α_1	Becomes:	<foo>→β_1<bar>
<foo> →<foo>α_2		<foo>→β_1<bar>
…		…
<foo> →<foo>α_n		<foo>→β_n<bar>
<foo>→β_1		
<foo>→β_2		<bar> →α_1<bar>
…		<bar> →α_2<bar>
<foo>→β_n		…
		<bar> →α_n<bar>
		<bar>→ε

In the case of our grammar, we map ***E*** to ***foo, abc*** to $\boldsymbol{\alpha_1}$, ***cde*** to $\boldsymbol{\alpha_2}$**,** and ***T*** to $\boldsymbol{\beta_1}$. Therefore, we obtain G′ as:

E→TE′
E′→abcE′
E′→cdeE′
E′→ε
T→bc

Note that E′ is a new non-terminal we introduced to eliminate left-recursion.

Exercise 2:

Consider the CFG G with non-terminals {S,A }, terminals {a,b}, start symbol S and productions:

S → Aa
S→b
A→Aab
A→ ε

Is the grammar left-recursive? If yes, eliminate the left recursion.

Solution:

Yes, the grammar is left-recursive with productions ***A→Ab | ε***

Based on the rule we have seen in class, we map ***A*** to ***foo, ab*** to $\boldsymbol{\alpha_1}$, ***and ε*** to $\boldsymbol{\beta_1}$. Therefore:

S → Aa
S→ b
A→A'
A'→abA'
A'→ ε

Note that A' is a new non-terminal we introduced to eliminate left-recursion.

Exercise 3:

Consider the CFG G with non-terminals {S,A }, terminals {a,b,c,d}, start symbol S and productions:

S → Aa
S→b
A→Sd
A→ ε

Is the grammar left-recursive? If yes, eliminate the left recursion.

Solution:

Yes, the grammar is left-recursive since it can be re-written as follows:

S → Aa
S→ b
A→Aad
A→bd
A→ ε

I have just replaced ***S*** by its value in ***A →Sd***. This shows that the grammar is clearly left-recursive with ***A →Aad|bd|ε***.

Based on the rule we have seen in class, we map ***A*** to ***foo, ad*** to α_1, ***bd*** to β_1, ***and ε*** to β_2. Therefore:

S → Aa
S→ b
A→A'
A→bdA'
A'→adA'
A'→ ε

Note that A' is a new non-terminal we introduced to eliminate left-recursion.

Exercise 4:

Consider the CFG G with non-terminal {S,T }, terminals {a,b,c,d,e}, start symbol S and productions:

S → abcd
  | abc
  | ab
  | Tabc
T→e
Perform left factoring on G.

Solution:

Let's review the rule we have seen in class:

For each non-terminal **A**, find the longest prefix α common to two or more of its alternatives

If $\alpha \neq \varepsilon$**, then replace all of the A productions**

A→ $\alpha\beta_1 | \alpha\beta_2 | \alpha\beta_3 | \ldots | \alpha\beta_n$

With

A→ α A'

A'→$\beta_1 | \beta_2 | \beta_3 | \ldots | \beta_n$

Hence, for ***S→abcd|abc|ab***, the longest common prefix is ***ab***. If we apply the transformation described in the rule above, we obtain:
S → abS'
  | Tabc
S'→cd
  |c
  |ε
T→e
Furthermore, we notice that we can still perform left factoring on ***S'→cd|c*** since ***c*** is a common prefix. Therefore, we obtain:
S → abS'
  | Tabc
S'→cS''
  |ε
S''→d
  | ε
T→e
Note that S' and S'' are new non-terminals we introduced to perform left factoring.

Exercise 5:

Consider the CFG G with non-terminal {S,E }, terminals {if,then,else,a,b}, start symbol S and productions:

S→if E then S
  | if E then S else S
  | a
E→b
Perform left factoring on G.

Solution:

S→if E then SS'
  | a
S'→else S
  | ε
E→b

Exercise 6:

Given the grammar:

<S> ::= <X><Y>

<X> ::= h | ε

<Y> ::= p

Where <S>, <X>, and <Y> are the non-terminal symbols.

Find the FIRST sets for all non-terminals.

Solution:

We refer to the rules covered in class:

1) FIRST(terminal)is{terminal}

2) If $A \rightarrow a\alpha$, and a is a terminal:

$$\{a\} \in FIRST(A)$$

3) If $A \rightarrow B\alpha$, and rule $B \rightarrow \varepsilon$ does **NOT** exist:

$$FIRST(B) \in FIRST(A)$$

4) If $A \rightarrow B\alpha$, and rule $B \rightarrow \varepsilon$ **DOES** exist:

$$\{(FIRST(B) - \varepsilon) \cup FIRST(\alpha)\} \in FIRST(A)$$

FIRST(X) = {h, ε} (2nd rule)

FIRST(Y) = {p} (2nd rule)

FIRST(S) = [FIRST(X)- ε] ∪ FIRST(Y) = {h, p} (4th rule)

Exercise 7:

Given the grammar:

<S> ::= <X><Y>

<X> ::= h

<Y> ::= p | ε

Where <S>, <X>, and <Y> are the non-terminal symbols.

Find the FOLLOW sets for all non-terminals.

Solution:

We refer to the rules covered in class:

> **1)**{\$}ϵFOLLOW(S)
> **2)** If A→ αB:
> FOLLOW(A) ϵ FOLLOW(B)
> **3)** If A→αBγ, and γ→ε does **NOT** exist:
> FIRST(γ) ϵ FOLLOW(B)
> **4)** If A→αBγ, and γ→ε **DOES** exist:
> {(FIRST(γ)-ε)υFOLLOW(A)}ϵFOLLOW(B)

FOLLOW(S) = {\$} (1st rule)

FOLLOW(X) = [FIRST(Y)- ε]υFOLLOW(S)

= {p, \$} (4th rule)

FOLLOW(Y) = FOLLOW (S)

= {\$} (2nd rule)

Exercise 8:

Given the grammar:
<S> ::= <X><Y>h
<X> ::= h | ε
<Y> ::= p | ε

Where <S>, <X>, and <Y> are the non-terminal symbols.

Find the FIRST and FOLLOW sets for all non-terminals.

Solution:

For the FIRST sets calculations:

FIRST(X)={h, ε} (2^{nd} rule … or just common sense!)

FIRST(Y)= {p, ε} (2^{nd} rule as well…)

Now, the FIRST(S) is the only challenging one.

I know that the 2^{nd} rule will not apply since <S> derives a string that does not start with a terminal.

How about the 3^{rd} rule? I will try to apply it by mapping the symbols of the 3^{rd} rules onto the symbols of the only production where <S> appears on the left-hand side (remember, when you are finding the FIRST for a terminal, you only look at the productions where that terminal appears on the left-hand side).

$$\overset{A}{\langle S\rangle} ::= \overset{B}{\langle X\rangle}\overbrace{\langle Y\rangle h}^{\alpha}$$

The mapping works great. However, the 3^{rd} rule specifically says that B$\rightarrow$ε should **NOT** exist. However, I do have a production X$\rightarrow$ε. Therefore, I cannot apply the 3^{rd} rule. So, let's analyze the 4^{th} rule (which has the same mapping as the 3^{rd} rule):

$$\overset{A}{\langle S\rangle} ::= \overset{B}{\langle X\rangle}\overbrace{\langle Y\rangle h}^{\alpha}$$

Mapping works great! Also, the 4^{th} rule says that B$\rightarrow$ ε should exist (which it does due to X$\rightarrow$ ε). So, I apply the rule as follows:

[(FIRST(X)- ε) υ FIRST(Yh)] є FIRST(S)

I know already what is the FIRST(X), but what about FIRST(Yh)? I do not have it yet and therefore, I need to calculate it. How do I calculate the FIRST of two symbols? That's simple, just create a production that summarizes the two symbols into one as follows:

temp→Yh

Therefore, if I find the FIRST(temp), I have the FIRST(Yh). Note that temp is just a temporary symbol I created to aid with this operation and will discard immediately after.

I know that I cannot find the FIRST(temp) using the 2nd rule because temp derives a string that does not start with a terminal. So it's either the 3rd or 4th rule.

The 3rd rule maps quite well:

$$\begin{array}{llll} A & & B & \alpha \\ \uparrow & & \uparrow & \uparrow \\ \text{temp} & \rightarrow & Y & h \end{array}$$

However, it is not applicable since Y→ε. But the 4th rule which has the same mapping works and so I apply it:

$[(\text{FIRST}(Y)- \varepsilon) \cup \text{FIRST}(h)] \in \text{FIRST}(\text{temp})$

I have the FIRST(Y). I also know that the FIRST(h) is simply {h} (see the 1st rule for the FIRST set calculation). Therefore:

FIRST(temp)={h,p}=FIRST(Yh)

Remember, I did all that work to arrive at the FIRST(S). I had the following relationship:

$[(\text{FIRST}(X)- \varepsilon) \cup \text{FIRST}(Yh)] \in \text{FIRST}(S)$

Since, I now know the FIRST(X) and FIRST(Yh), I can populate the FIRST(S) set:

FIRST(S)={h,p}

For the FOLLOW sets calculation:

FOLLOW(S)={$} (using the 1st rule).

To obtain the FOLLOW(X), I need to analyze the following production (since it's the only one where X appears on the right-hand side):

<S> ::= <X><Y>h

The 2nd rule does not work since X does not appear at the end of the string XYh. However, the 3rd rules maps well as shown below:

$$\overset{A}{S} \rightarrow \overset{\alpha}{\varepsilon}\ \overset{B}{X}\ \overbrace{Y\ h}^{\gamma}$$

The 3rd rule states that $\gamma \rightarrow \varepsilon$ does not exist. However, γ refers to two symbols instead of one. Therefore, the only way $\gamma \rightarrow \varepsilon$ would exist is if $Y \rightarrow \varepsilon$ **and** $h \rightarrow \varepsilon$. While Y does derive epsilon, h cannot derive anything other than itself, since it is a terminal. Therefore, you will never have a situation where $\gamma \rightarrow \varepsilon$. I can safely conclude that $\gamma \rightarrow \varepsilon$ does not exist. Hence, the 3rd rule is applicable as follows:

FIRST(Yh) $\in$ FOLLOW(X)

We already have FIRST(Yh). Therefore:
FOLLOW(X)={h,p}

To obtain FOLLOW(Y), I also simply apply the 3rd rule which yields:

FIRST(h) $\in$ FOLLOW(Y)

This means that:

FOLLOW(Y)={h}