After class, I will
1) Assign HW 5 (will be due next Fri)
2) Scan notes for this week.

Modifying grammars to equivalent (nicer) grammars.

Last time: removing useless productions.

Today: \( \lambda \)-productions and unit productions
\((A \rightarrow \lambda)\)
\((A \rightarrow B)\)

How do we "get rid of" \( \lambda \)-productions?

If we have
\[
A \rightarrow aByC, \\
B \rightarrow \lambda \\
C \rightarrow yxBaA
\]
we add prods.

\[
A \rightarrow aBycC | ayC C \\
B \rightarrow yC \\
C \rightarrow yxBaA \mid yxaA
\]
and now we can remove \( B \Rightarrow \lambda \).
We could have a more complicated situation:

\[ A \rightarrow BC \]

\[ B \rightarrow \lambda \]

\[ C \rightarrow \lambda \]

In this situation, we can also produce \( \lambda \) from \( A \) (by replacing \( A \) w/\( BC \) and then both \( B \) + \( C \) with \( \lambda \)'s).

It's possible we might worry this process gets us in an infinite loop, let's try to do all our changes at once.

1) Find all vars that can go to \( \lambda \) (nullable vars)

(anything with \( X \rightarrow \lambda \) is nullable.

inghting with \( X \rightarrow \lambda \) all nullable vars is nullable)
2) In any production that contains a nullable variable, add every possibility of deleting a nullable var as a new production.

\[ \text{e.g. If } B \text{ and } C \text{ are nullable} \]

\[ A \to xByCzB \text{ becomes } \]

\[ A \to xByCzB \mid xyCzB \mid xByzB \mid xByCz \mid \]

\[ xyB \mid xyCz \mid xByz \mid xyz \]

So if a prod. has \( k \) nullable variables, it is replaced with \( 2^k \) productions.

3) Then it is safe to delete all the \( \lambda \)-prods - they have all been replaced.
Removing unit productions:

E.g. \[ A \rightarrow B / x y C \]

\[ B \rightarrow x y A B C | C x z A | x z B \]

We can replace \( A \rightarrow B \) with

\( A \rightarrow \) all the things \( B \) goes to

so we get

\[ A \rightarrow x y C | x y A B C | C x z A | x z B \]

\[ B \rightarrow x y A B C | C x z A | x z B \]

There are complications; we might have

\[ A \rightarrow B | \ldots \]

\[ B \rightarrow C | \ldots \]

\[ C \rightarrow A | \ldots \]
To take care of this we draw a directed graph (nodes = vars, arrows = unit prods).

Replace every var:

1. Add, as productions for A, all prods for variables one can get to from A in this graph.

2. Now it's safe to delete all unit productions.

In this the above case, one can combine A, B, C into one var, but we can have

where D still needs to be a separate var.
Notes: if we act in the right order, we can remove all useless prods, all \( \lambda \)-prods, and all unit prods.

---

**Chomsky Normal Form (CNF)**

A grammar is in CNF if all the productions look like

\[
A \rightarrow BC \quad (\text{one var} \rightarrow 2 \text{ vars}) \\
\quad (A \rightarrow AA \text{ or } A \rightarrow AB \text{ okay}) \\
\quad \text{or } A \rightarrow x \quad (\text{one var} \rightarrow \text{one letter})
\]

---

Turning grammars to CNF:

E.g., \( A = \text{start var:} \)

\[
\begin{align*}
A & \rightarrow xyAB \mid Bx \mid y \\
B & \rightarrow xByB \mid BA \mid x
\end{align*}
\]

\[\text{0) Get rid of useless, } \lambda, + \text{ unit prods}\]
1. Introduce a var for each letter
   \[ X \rightarrow x \]
   \[ Y \rightarrow y \]
   \[ A \rightarrow XYAB |BX|y \]
   \[ B \rightarrow XBYB |BAX|x \]

2. Introduce new "glued variables"
   \[ AX \rightarrow AX \]
   \[ AB \rightarrow AB \]
   \[ YB \rightarrow YB \]
   \[ BB \rightarrow BYB \]

make this readable:

\[
\begin{align*}
C &= [AX] \\
D &= [YAB] \\
E &= [BYB] \\
F &= [AB] \\
G &= [YB]
\end{align*}
\]

\[
\begin{align*}
C &\rightarrow AX \\
D &\rightarrow YF \\
E &\rightarrow BG \\
F &\rightarrow AB \\
G &\rightarrow YB \\
A &\rightarrow XD |BX|y \\
B &\rightarrow XE |BC|x \\
X &\rightarrow x \\
Y &\rightarrow y
\end{align*}
\]
Greibach Normal Form

All productions are of the form

\[ A \to \times \text{STUFF} \]

(where STUFF could be \( \lambda \) - you can insist STUFF is all variables)

We won't talk about how to convert to GNF, but it is possible.