Goal: For every right linear grammar $G$, there is an NFA $M$ so that the language generated by $G$ is the same as the language accepted by $M$, and vice versa (i.e. for every NFA $M$, there is a right-linear gram $G$, ...)

Reminder: A right-linear grammar is one where all productions look like:

$A \rightarrow wB$ or $A \rightarrow w$

single variables strings in $\Gamma^*$
Example:

\[ G \text{ is the grammar} \]

\[
\begin{align*}
S & \rightarrow aba A \mid bbB \mid abbc \mid b \\
A & \rightarrow baB \mid bc \mid a \\
B & \rightarrow bbA \mid aS \mid aB \mid baC \\
C & \rightarrow bbab \mid abA
\end{align*}
\]

Make our NFA as follows
Example of string generated by $G$

$$S \rightarrow bbB \rightarrow bbbac \rightarrow bbbabaA$$
$$\rightarrow bbbaba$$

This gives a path through our NFA accepting the string $bbbaaba$.

Our NFA is constructed so that a path to the final state $B$ corresponds to a string $w$ in our grammar.
Formalize this:

Given a right-linear grammar \( G = (V, \Gamma, S, P) \), we construct an NFA \( M = (Q, \Gamma, S, q_0, F) \) by

\[
Q = V \cup \Theta
\]

\[
\Gamma' = \Gamma
\]

\[
s = (Q \times \Gamma^*) \times Q
\]

\[
q_0 = S
\]

\[
F = \{f\}
\]

\( S \) is given by:

for each production \( V \rightarrow wV' \) in \( P \), we have

\[
((V, w), V') \in S
\]

and for each production \( V \rightarrow w \) in \( P \), we have

\[
((V, w), f) \in S
\]
How do we start w/ an NFA and make a right-linear grammar?

Example:

We should reverse what we just did:
Variables are $A, B, C, D$

$A \rightarrow abB \mid aC$

$B \rightarrow bA$

$C \rightarrow baD \mid A \mid B \mid \lambda$

$D \rightarrow bB \mid aC \mid \lambda$
Example string: \(aba\) is accepted since we have

So \(aba\) is generated by our grammar by

\[
A \rightarrow a C \rightarrow aB \rightarrow abA \rightarrow abaC \rightarrow aba
\]

Summary of course so far:

Def'n: A **regular language** is a language \(L\) for which there is a PFA \(M\) accepting \(L\).

We have proven: A regular language can also be described as

1) The set of strings accepted by an NFA
2) The set of strings matching a regular expression
3) The set of strings generated by a right-linear grammar.
Next time: Operations on regular languages that make new regular languages (e.g. truncate)

Practice: Make a regex from an NFA

Remove B:

\[
\begin{align*}
A & \rightarrow a + ab^*(a+\lambda) \\
B & \rightarrow b + bb^*(a+\lambda) \\
C & \rightarrow bb^*(b+\lambda)
\end{align*}
\]
The regular expression is:

\[(a+ab^*(b+\lambda))^* (b+ab^*(a+\lambda))\]

\[
\left[ (b+bb^*(a+\lambda)) + bb^*(b+\lambda)(a+ab^*(b+\lambda))^* (b+ab^*(a+\lambda)) \right]^* \]