~~Regul~~ Right-Linear Grammars & ~~DFAs~~ NFAs

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-lin. gram G, ----)

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

$$A \longrightarrow wB \quad \text{or} \quad A \longrightarrow w$$

single variables   strings in T^*

Example:

G is the grammar

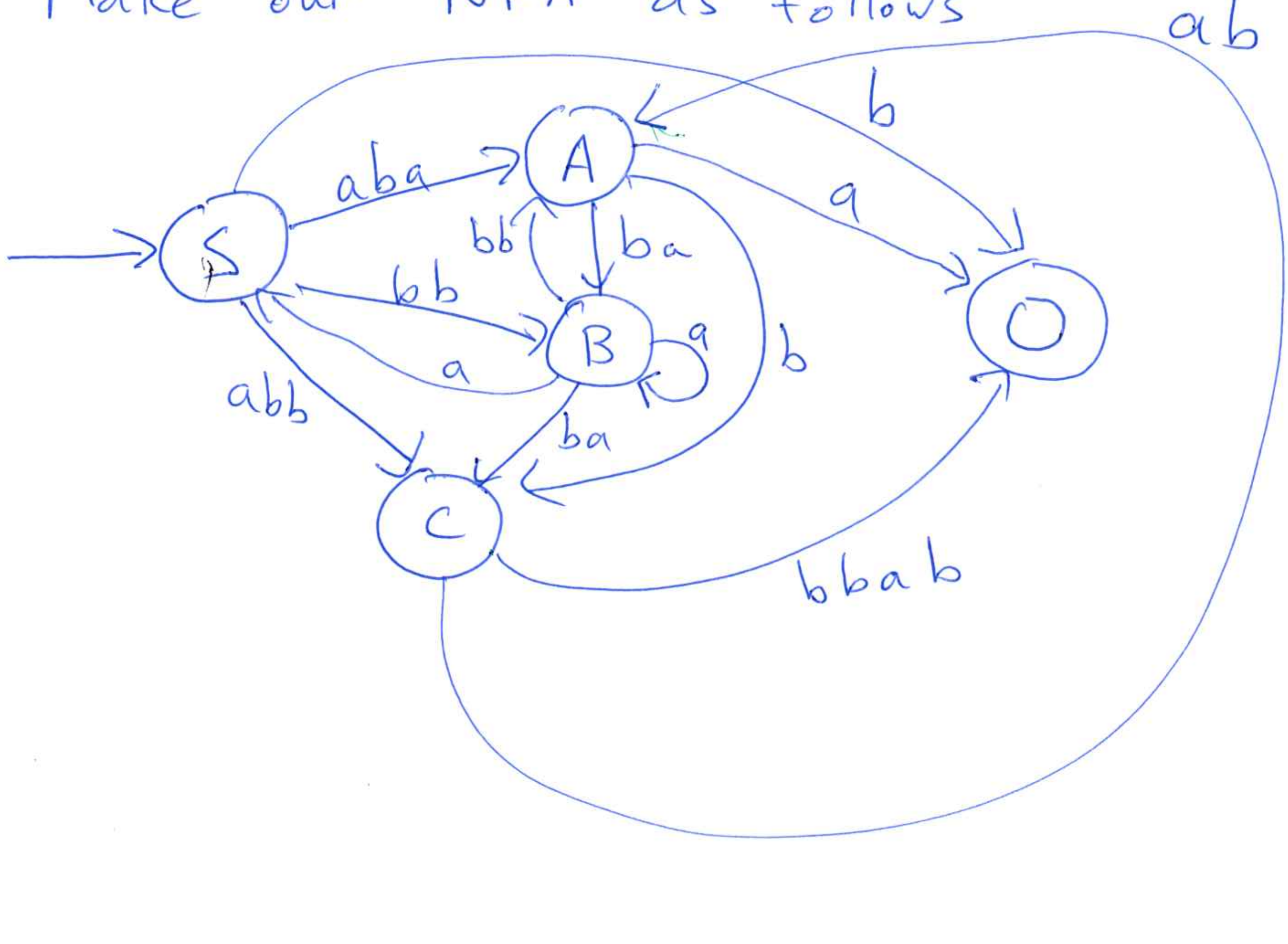
$$S \rightarrow abaA \mid bbB \mid abbC \mid b$$

$$A \rightarrow baB \mid bC \mid a$$

$$B \rightarrow bbA \mid aS \mid aB \mid baC$$

$$C \rightarrow bbab \mid abA$$

Make our NFA as follows



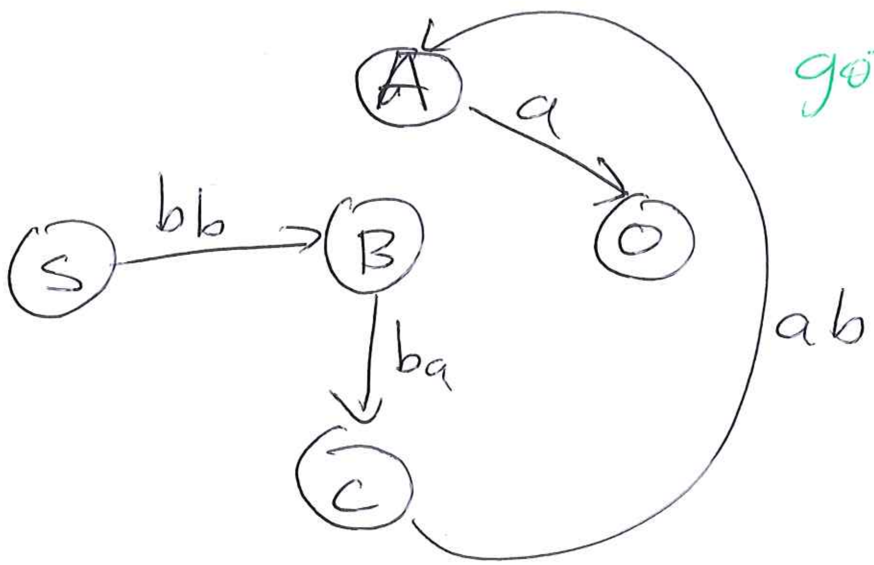
Example of string generated by G

$S \to bbB \longrightarrow bbbaC \longrightarrow bbbaabA$

$\longrightarrow bbbaaba$

This gives a path through our NFA accepting the string bbbaaba

going to B



Our NFA is constructed so that a path to the final state ~~is~~ ~~corresponds~~ that processes a string w corresponds to a way to generate w in our grammar.

Formalize this:

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

$Q = V \cup \{f\}$

$T = T$

$\delta \subseteq (Q \times T^*) \times Q$

$q_0 = S$

$F = \{f\}$

δ is given by:

for each production $V \longrightarrow wV'$ in P, we have

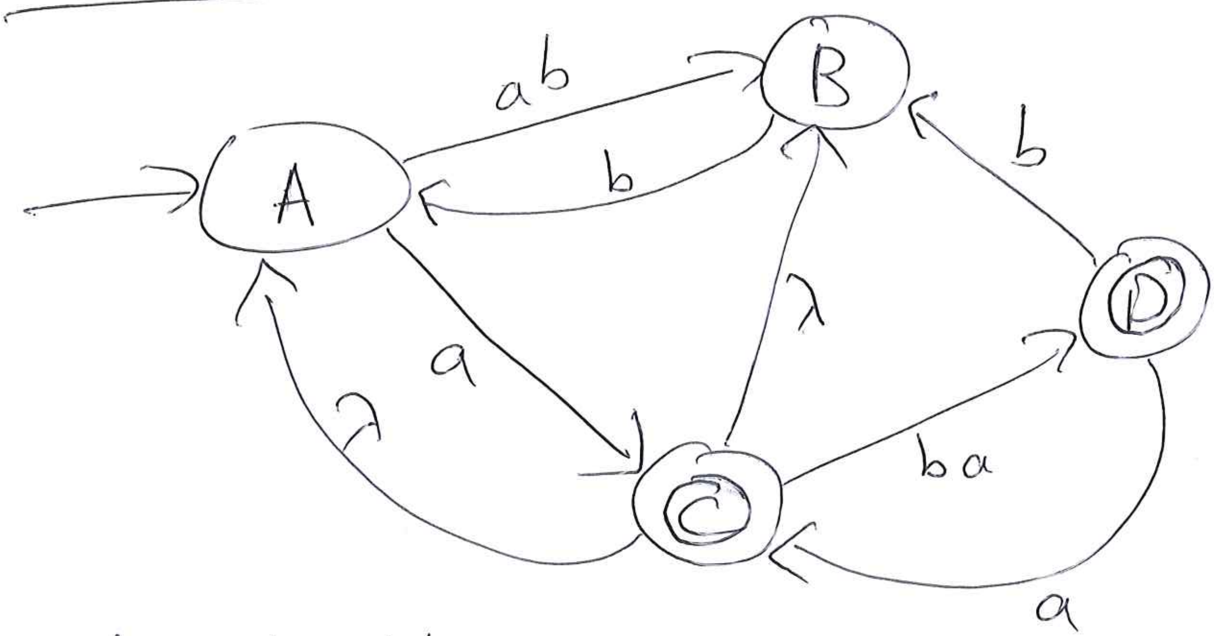
$$((V,w),V') \in \delta.$$

and for each production $V \longrightarrow w$ in P, we have

$$((V,w),f) \in \delta.$$

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

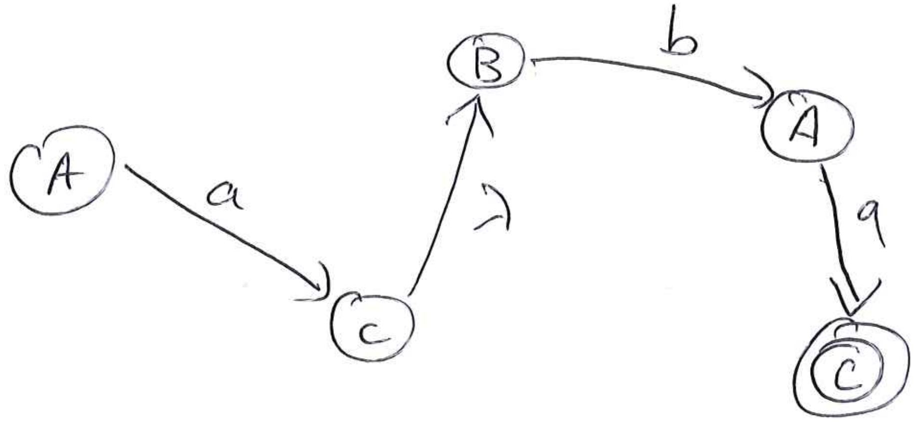
$A \rightarrow abB \mid aC$

$B \rightarrow bA$

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

$D \rightarrow bB \mid aC \mid \lambda$

Example string: aba is accepted since we have



in NFA

So aba is generated by our grammar by

$$A \to aC \to aB \to abA \to abaC \to aba$$

Summary of course so far:

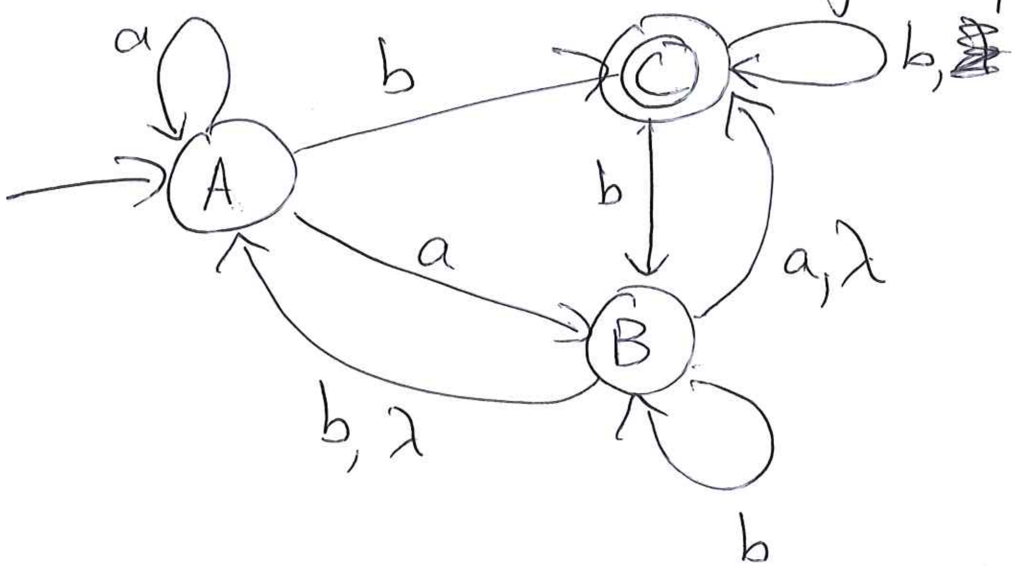
<u>Def'n</u>: A <u>regular</u> language is a language L for which there is a DFA M accepting L.

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 reg exp.
3) The set of strings generated by a right-linear grammar.

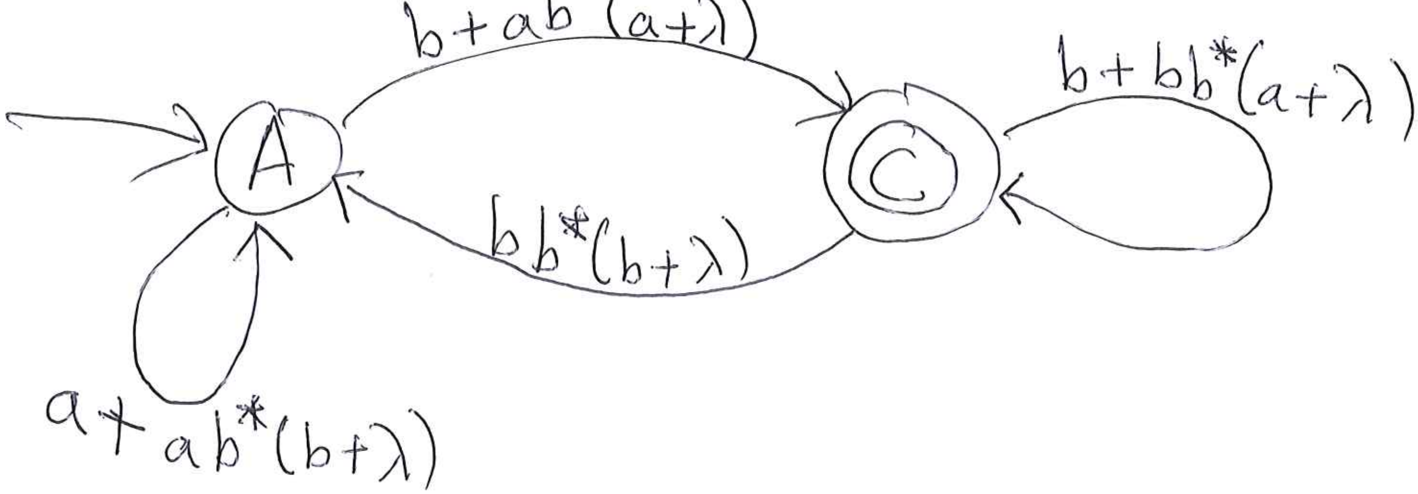
Next time: ① Operations on ~~languag~~ regular languages that make new regular languages (e.g. truncate)

<u>Practice</u>: Make a reg exp from an NFA



---

remove B:



The regular expression is:

$$(a+ab^*(b+\lambda))^*(b+ab^*(a+\lambda))$$

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