Note on regular grammars

The purpose of this note is to point out a neat trick for converting certain context-free grammars into finite automata, and vice-versa.

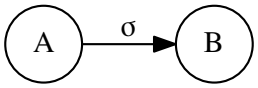
Definition 1. *A context-free grammar is* **regular** *if any occurrence of a variable on the right-hand side of a rule is as the rightmost symbol.*

Notice that this means that we only ever have one choice of which variable to expand, and also that this means the generating rules of the grammar look a *lot* like transitions in a finite automaton.

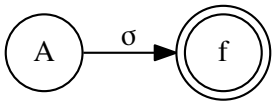
The rules for making this conversion are straightforward.

To convert a regular grammar to a machine:

- each variable becomes a state
- each rule $A \to \sigma B$ (where A and B are variables, and $\sigma \in \Sigma \cup \{\varepsilon\}$ becomes

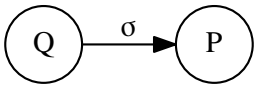


- each rule $A \to \sigma$ (where A and B are variables, and $\sigma \in \Sigma \cup \{\varepsilon\}$ becomes



To convert a finite automaton to a regular grammar

- each state becomes a variable
- each transition $\delta(q, \sigma) = p$ for $\sigma \in \Sigma \cup \{\varepsilon\}$



becomes the rule

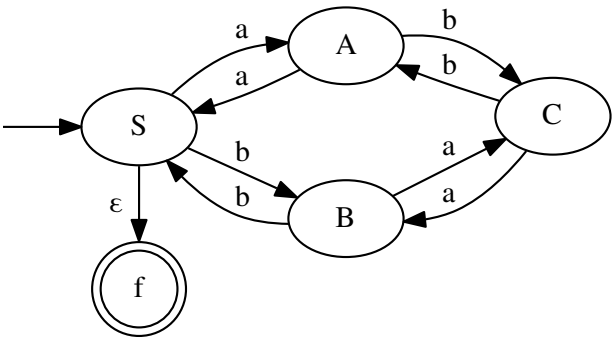
$$Q \to \sigma P$$

- if state q is final, add a rule $Q \to \varepsilon$

An example regular context-free grammar:

$$S \to aA \mid bB \mid \varepsilon$$
$$A \to aS \mid bC$$
$$B \to bS \mid aC$$
$$C \to aB \mid bA$$

An automaton recognizing the same language:



You should try practicing this trick on automata we have already seen. You should also try writing a regular grammar and then converting it to an automaton!