CS3110 Formal Language and Automata

Tingting Chen
Computer Science
Cal Poly Pomona

Grammar

- A grammar $G = (V, T, S, P)$ consists of the following quadruple:
  - a set V of variables (non-terminal symbols), including a starting symbol $S \in NT$
  - a set T of terminals (same as an alphabet, Σ)
  - A start symbol $S \in V$
  - a set P of production rules
- Example:

  $S \rightarrow aS \mid A$

  $A \rightarrow bA \mid \lambda$

Derivation

- Strings are "derived" from a grammar
- Example of a derivation

  $S \Rightarrow aS \Rightarrow aaS \Rightarrow aaA \Rightarrow aabA \Rightarrow aab$
- At each step, a nonterminal is replaced by the **sentential form** on the right-hand side of a rule (a sentential form can contain nonterminals and/or terminals)
- Automata *recognize* languages; grammars *generate* languages

Context-free grammar

- A grammar is said to be context-free if every rule has a single non-terminal on the left-hand side
- This means you can apply the rule in any context. More complicated languages (such as English) have context-dependent rules. A language generated from a context-free grammar is called a context-free language

Regular grammar

- A grammar is said to be **right-linear** if all productions are of the form $A \to xB$ or $A \to x$, where A and B are variables and x is a string of terminals
- A grammar is said to be **left-linear** if all productions are of the form $A \to Bx$ or $A \to x$
- A regular grammar is either right-linear or left-linear.

Linear grammar

- A grammar can be linear without being right- or left-linear.
- A linear grammar is a grammar in which at most one variable can occur on the right side of any production rule, without any restriction on the position of the variable.
- Example:

$S \rightarrow aS \mid A$

$A \rightarrow Ab \mid \lambda$

Another formalism for regular languages

- Every regular grammar generates a regular language, and every regular language can be generated by a regular grammar.
- A regular grammar is a simpler, special-case of a context-free grammar
- The regular languages are a proper subset of the context-free languages

Exercise

- Given a grammar, you should be able to say what language it generates
- Use set notation to define the language generated by the following grammars

1) $S \rightarrow aaSB \mid \lambda$
   $B \rightarrow bB \mid b$

2) $S \rightarrow aSbb \mid A$
   $A \rightarrow cA \mid c$

Exercise

$S \to aaSB \mid \lambda$
$B \to bB \mid b$

It helps to list some of the strings that can be formed:

$S \Rightarrow aaSB \Rightarrow aaB \Rightarrow aab$

$S \Rightarrow aaSB \Rightarrow aaB \Rightarrow aabB \Rightarrow aabb$

$S \Rightarrow aaSB \Rightarrow aaB \Rightarrow aabB \Rightarrow aabbB \Rightarrow aabbb$

$S \Rightarrow aaSB \Rightarrow aaB \Rightarrow aabB \Rightarrow aabbB \Rightarrow aabbbB \Rightarrow aabbbb$

$S \Rightarrow aaSB \Rightarrow aaaaSBB \Rightarrow aaaaBB \Rightarrow aaaaBb \Rightarrow aaaabb$

$S \Rightarrow aaSB \Rightarrow aaaaSBB \Rightarrow aaaaBB \Rightarrow aaaaBbB \Rightarrow aaaaBbb \Rightarrow aaaabbb$

What is the pattern?

$L = \{(aa)^n b^n b^m\}$

Exercise

- Given a language, you should be able to give a grammar that generates it.
- For example, give a regular (right-linear) grammar for the language consisting of all strings over {a, b, c} that begin with a, contain exactly two b's, and end with cc.

Exercise

- Give a regular (right-linear) grammar for the language consisting of all strings over {a, b, c} that begin with a, contain exactly two b's, and end with cc

  $S \rightarrow aA$

  $A \rightarrow bB \mid aA \mid cA$

  $B \rightarrow bC \mid aB \mid cB$

  $C \rightarrow aC \mid cC \mid cD$

  $D \rightarrow c$

Theorem

- **Every language generated by a right-linear grammar is regular.**
- *Proof:*
  - Specify a procedure for automatically constructing an NFA that mimics the derivations of a right-linear grammar.

Theorem– Right linear grammar to FA

- Justification:
  - The sentential forms produced by a right linear grammar have exactly one variable, which occurs as the rightmost symbol.
  - Assume that our grammar has a production rule
    $D \rightarrow dE$
    and that, during the derivation of a string, there is a step
    $wcD \Rightarrow wcdE$
  - We can construct an NFA which has states D and E, and an edge labeled d from D to E.
  - NFAs can be converted to DFAs.
  - All languages accepted by DFAs are regular.

Theorem– Right linear grammar to FA

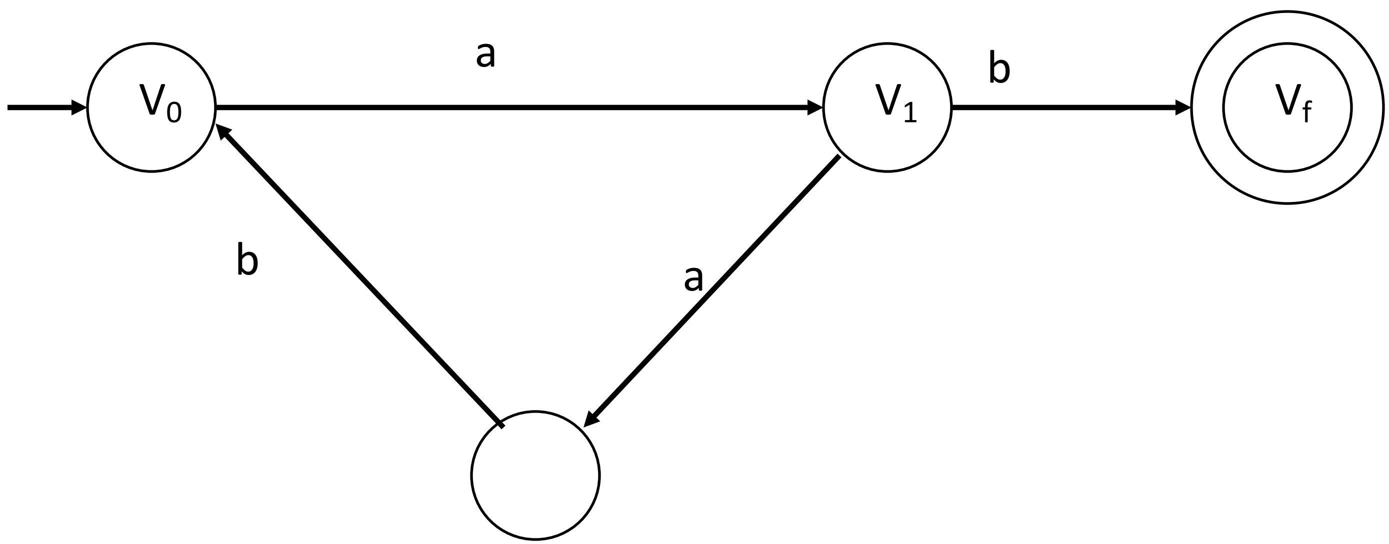
- Construction:
  - For each variable V_i in the grammar there will be a state in the automaton labeled V_i.
  - The initial state of the automaton will be labeled V_0 and will correspond to the S variable in the grammar.
  - For each production rule $V_i \to a_1a_2\ldots a_mV_j$ the automaton will have transitions such that
    $\delta^*(V_i , a_1a_2\ldots a_m) = V_j$
  - For each production rule $V_i \to a_1a_2\ldots a_m$ the automaton will have transitions such that
    $\delta^*(V_i , a_1a_2\ldots a_m) = V_{final}$

Right linear grammar to FA -- Example

Construct an NFA that accepts the language generated by the grammar:

$S \rightarrow aA$ convert to: $V_0 \rightarrow aV_1$

$A \rightarrow abS \mid b$ $V_1 \rightarrow abV_0 \mid b$



Right linear grammar to FA -- Exercise

Construct an NFA that accepts the language generated by the grammar:

$S \to aA$

$A \to abS \mid bA \mid b$

Theorem: DFA to right-linear grammar

- **Every regular language can be generated by a right-linear grammar.**
- *Proof:*
  - Generate a DFA for the language.
  - Specify a procedure for automatically constructing a right-linear grammar from the DFA.

Theorem: DFA to right-linear grammar

- Given a regular language L, let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA that accepts L. Let $Q = \{q_0, q_1, \ldots, q_n\}$ and $\Sigma = \{a_1, a_2, \ldots, a_m\}$.
- Construct the grammar $G = (V, T, S, P)$ with:
  $V = \{q_0, q_1, \ldots, q_n\}$
  $T = \{a_1, a_2, \ldots, a_m\}$
  $S = q_0$.
  $P = \{\}$ initially.
- P, the set of production rules, is constructed as follows:

Theorem: DFA to right-linear grammar

- For each transition of M

  $\delta(q_i, a_j) = q_k$

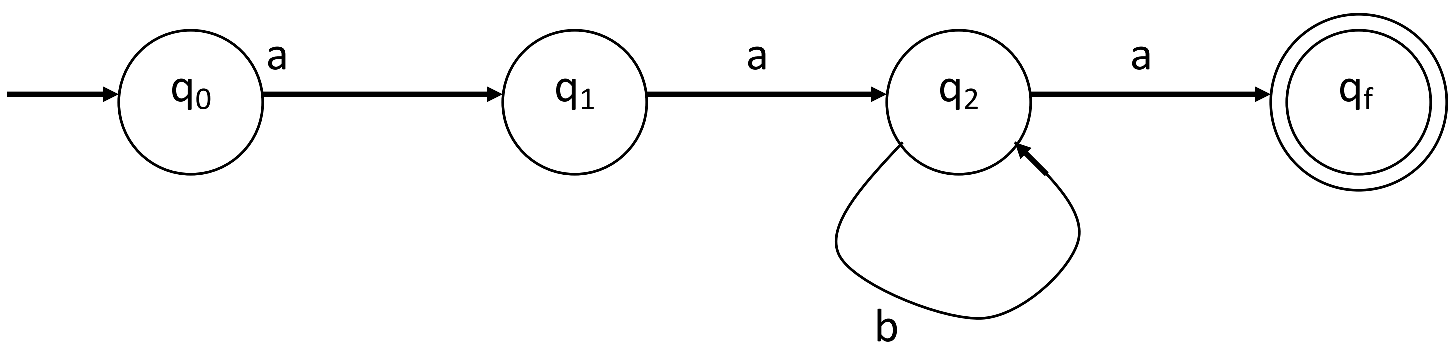
  add to P the production:

  $q_i \rightarrow a_j q_k$
- If q_k is in F, then add to P the production:

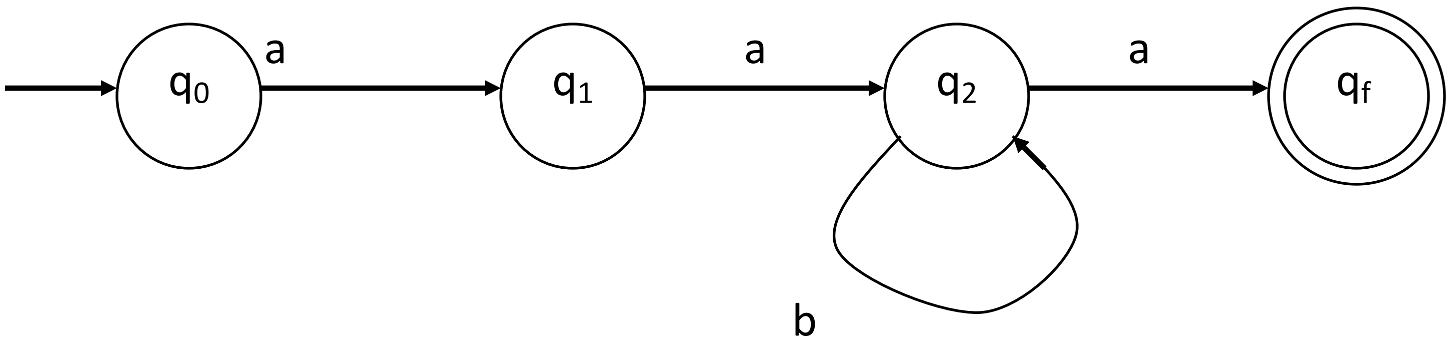
  $q_k \rightarrow \lambda$

DFA to right-linear grammar

- Example: Construct a right-linear grammar for the language L = L(aab^*a)
- First, build an NFA for L:



DFA to right-linear grammar: Example



P = {} initially.

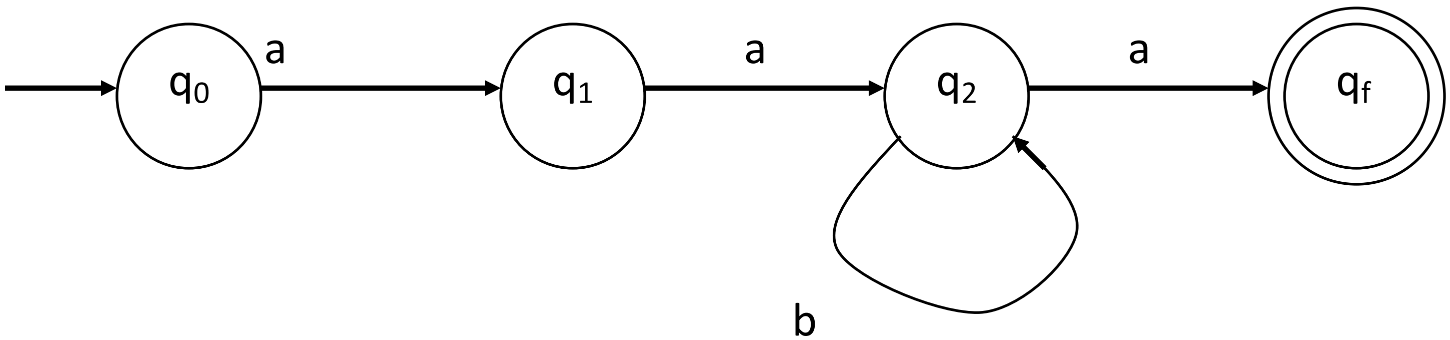
Add to P a rule for each transition in the NFA:

$$q_0 \rightarrow aq_1$$
$$q_1 \rightarrow aq_2$$
$$q_2 \rightarrow bq_2$$
$$q_2 \rightarrow aq_f$$

Since q_f is in F, add to P the production:

$$q_f \rightarrow \lambda$$

DFA to right-linear grammar: Example



Now P =

$\{q_0 \to aq_1$

$q_1 \to aq_2$

$q_2 \to bq_2$

$q_2 \to aq_f$

$q_f \to \lambda\ \}$

You can convert to normal grammar notation:

$S \to aA$

$A \to aB$

$B \to bB$

$B \to aC$

$C \to \lambda$

Theorem: Left-linear grammar

A language L is regular if and only if there exists a left-linear grammar G such that L = L(G).

Proof:
The strategy here is a little tricky.
We first show the reverse of L, L^R is regular, by showing L^R can be generated by a right-linear grammar.

We describe an algorithm to construct a right-linear grammar that generates the reverse of all the strings generated by the left-linear grammar.

Theorem: Left-linear grammar

Given any left-linear grammar we can construct from it an right-linear grammar G' by replacing productions of the form:

$A \rightarrow Bv$ with $A \rightarrow v^R B$

and

$A \rightarrow v$ with $A \rightarrow v^R$

Since L(G') is generated by a right-linear grammar, it is regular.

It can be demonstrated that $L(G) = (L(G'))^R$.

It can be proven that the reverse of any regular language is also regular.

Hence, L is regular.

Theorem

A language L is regular if and only if there exists a regular grammar G such that L = L(G).

Proof:

Combine our definition of regular grammars, which includes the statement, “A regular grammar is either right-linear or left-linear”, with theorems 3.4 and 3.5

3 ways of specifying regular languages