Right-Linear Grammars

- All productions have form:

$$A \to xB$$

or

$$A \to x$$

string of terminals

- Example: $S \to abS$

$S \to a$

Left-Linear Grammars

- All productions have form:

$$A \to Bx$$

or

$$A \to x$$

string of terminals

- Example: $S \to Aab$

$A \to Aab \mid B$

$B \to a$

Regular Grammars

Regular Grammars

- A regular grammar is any right-linear or left-linear grammar

- Examples:

$S \to abS$	$S \to Aab$
$S \to a$	$A \to Aab \mid B$
	$B \to a$

What languages are generated by these grammars?

Languages and Grammars

$$S \to abS$$
$$S \to a$$

$$S \to Aab$$
$$A \to Aab \mid B$$
$$B \to a$$

$L(G_1) = (ab)^* a$ $\quad$ $L(G_2) = aab(ab)^*$

Note both these languages are regular
- we have regular expressions for these languages (above)
- we can convert a regular expression into an NFA (how?)
- we can convert an NFA into a DFA (how?)
- we can convert a DFA into a regular expression (how?)

Do regular grammars also describe regular languages??

Regular Grammars Generate Regular Languages

Theorem

$$\left\{\begin{array}{l}\text{Languages}\\\text{Generated by}\\\text{Regular Grammars}\end{array}\right\} = \left\{\begin{array}{l}\text{Regular}\\\text{Languages}\end{array}\right\}$$

Theorem - Part 1

$$\left\{\begin{array}{l}\text{Languages}\\\text{Generated by}\\\text{Regular Grammars}\end{array}\right\} \subseteq \left\{\begin{array}{l}\text{Regular}\\\text{Languages}\end{array}\right\}$$

Any regular grammar generates a regular language

Proof – Part 1

$$\left\{\begin{array}{l}\text{Languages}\\ \text{Generated by}\\ \text{Regular Grammars}\end{array}\right\} \subseteq \left\{\begin{array}{l}\text{Regular}\\ \text{Languages}\end{array}\right\}$$

The language $L(G)$ generated by any regular grammar G is regular

The case of Right-Linear Grammars

- Let G be a right-linear grammar
- We will prove: $L(G)$ is regular
- **Proof idea:** We will construct NFA using the grammar transitions

Example

Given right linear grammar:

$$V_0 \rightarrow aV_1$$
$$V_1 \rightarrow abV_0 | b$$

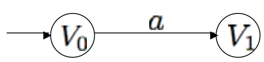
Step 1: Create States for Each Variable

- Construct NFA M such that every state is a grammar variable:

$$V_0 \rightarrow aV_1$$
$$V_1 \rightarrow abV_0 | b$$

Step 2.1:
Edges for Productions

- Productions of the form $V_i \rightarrow aV_j$ result in $\delta(V_i, a) = V_j$

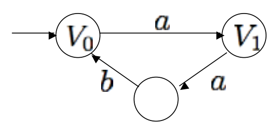


$V_0 \rightarrow aV_1$

$V_1 \rightarrow abV_0|b$

Step 2.2:
Edges for Productions

- Productions of the form $V_i \rightarrow wV_j$ are only slightly harder…. Create row of states that derive w and end in V_j

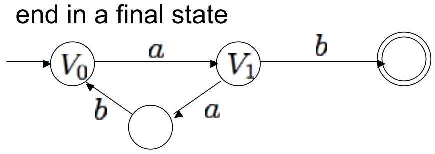


$V_0 \rightarrow aV_1$

$V_1 \rightarrow abV_0|b$

Step 2.3:
Edges for Productions

- Productions of the form $V_i \rightarrow w$ Create row of states that derive w and end in a final state



$V_0 \rightarrow aV_1$

$V_1 \rightarrow abV_0|b$

In General

- Given any right-linear grammar, the previous procedure produces an NFA
  - We sketched a proof by construction
  - Result is both a proof and an algorithm
  - *Why doesn't this work for a non linear grammar?*
- Since we have an NFA for the language, the right-linear grammar produces a regular language

Proof - Part 2

$$\left\{\begin{array}{l}\text{Regular}\\\text{Languages}\end{array}\right\} \subseteq \left\{\begin{array}{l}\text{Languages}\\\text{Generated by}\\\text{Regular Grammars}\end{array}\right\}$$

Any regular language L is generated by some regular grammar G

Any regular language L is generated by some regular grammar G

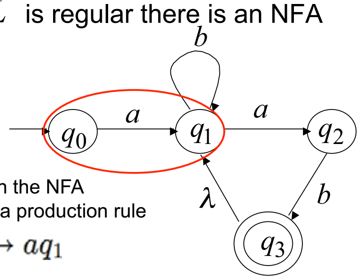
Proof idea:

Let M be the NFA with $L = L(M)$.

Construct from M a regular grammar G such that $L(M) = L(G)$

NFA to Grammar Example

- Since L is regular there is an NFA



This transition in the NFA
Looks a lot like a production rule

$q_0 \rightarrow aq_1$

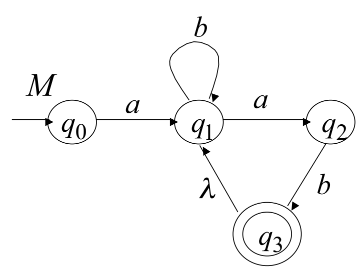
Step 1: Convert Edges to Productions

$q_0 \rightarrow aq_1$

$q_1 \rightarrow bq_1$

$q_1 \rightarrow aq_2$

$q_2 \rightarrow bq_3$



Step 2: λ Edges and Final States

$q_0 \to aq_1$

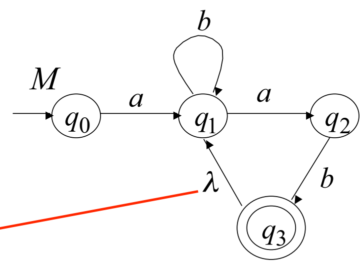
$q_1 \to bq_1$

$q_1 \to aq_2$

$q_2 \to bq_3$

$q_3 \to q_1$

$q_3 \to \lambda$



Step 2: λ Edges and Final States

$q_0 \to aq_1$

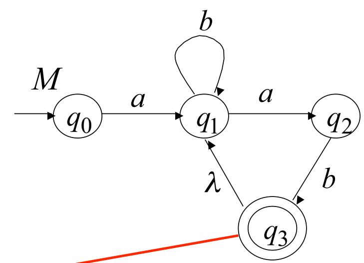
$q_1 \to bq_1$

$q_1 \to aq_2$

$q_2 \to bq_3$

$q_3 \to q_1$

$q_3 \to \lambda$



If q_i is a final state, add $q_i \to \lambda$

In General

- Given any NFA, the previous procedure produces a right linear grammar
  - We sketched a proof by construction
  - Result is both a proof and an algorithm
- Every regular language has an NFA
  - Can convert that NFA into a right linear grammar
  - Thus every regular language has a right linear grammar
- Combined with Part 1, we have shown right linear grammars are yet another way to describe regular languages

But What About Left-Linear Grammars

- What happens if we reverse a left linear grammar as follows:

$V_i \to V_j w$ Reverses to $V_i \to w^R V_j$

$V_i \to w$ Reverses to $V_i \to w^R$

- The result is a right linear grammar.
  - If the left linear grammar produced L, then what does the resulting right linear grammar produce?

But What About Left-Linear Grammars

- The previous slide reversed the language!

$V_i \to V_j w$ Reverses to $V_i \to w^R V_j$

$V_i \to w$ Reverses to $V_i \to w^R$

- If the left linear grammar produced language L, then the resulting right linear grammar produces L^R

Claim we just proved left linear grammars produce regular languages? Why?

Left-Linear Grammars Produce Regular Languages

- Start with a Left Linear grammar that produces L want to show L is regular
- Can produce a right linear grammar that produces L^R
- All right linear grammars produce regular languages so L^R is a regular language
- The reverse of a regular language is regular so

$(L^R)^R = L$ is a regular language!

For regular languages L_1 and L_2 we will prove that:

$$\left.\begin{array}{rl} \text{Union:} & L_1 \cup L_2 \\ \text{Concatenation:} & L_1 L_2 \\ \text{Star:} & L_1 * \\ \text{Reversal:} & L_1{}^R \\ \text{Complement:} & \overline{L_1} \\ \text{Intersection:} & L_1 \cap L_2 \end{array}\right\} \text{Are regular Languages}$$

We say: Regular languages are **closed under**

Union: $L_1 \cup L_2$

Concatenation: $L_1 L_2$

Star: $L_1 *$

Reversal: $L_1{}^R$

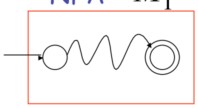
Complement: $\overline{L_1}$

Intersection: $L_1 \cap L_2$

Regular language L_1

$L(M_1) = L_1$

NFA M_1

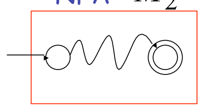


Single final state

Regular language L_2

$L(M_2) = L_2$

NFA M_2

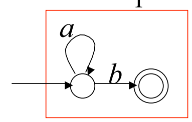


Single final state

Example

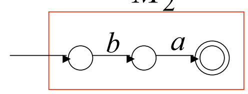
M_1

$L_1 = \{a^n b\}_{n \geq 0}$



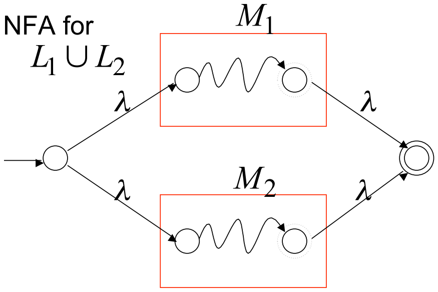
M_2

$L_2 = \{ba\}$



Union

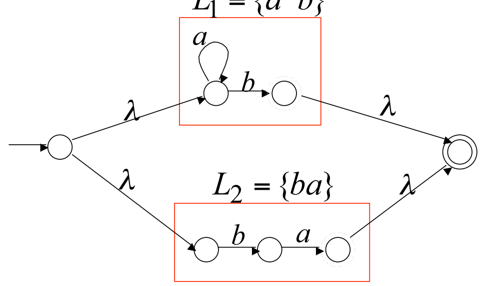
- NFA for $L_1 \cup L_2$



Example

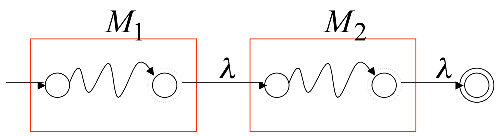
NFA for $L_1 \cup L_2 = \{a^n b\} \cup \{ba\}$

$L_1 = \{a^n b\}$



Concatenation

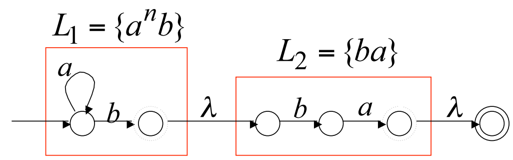
NFA for $L_1 L_2$



Example

$L_1 L_2 = \{a^n b\}\{ba\} = \{a^n bba\}$

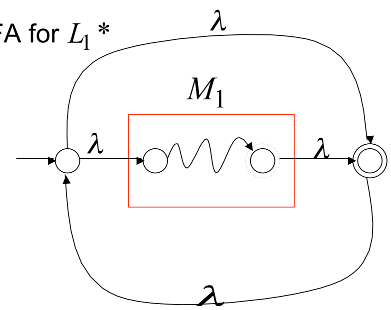
NFA for



Star Operation

NFA for $L_1{}^*$

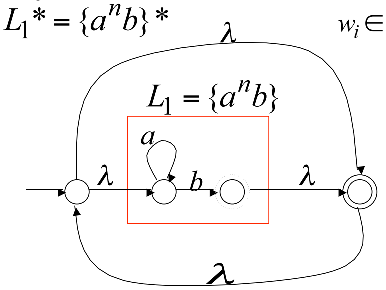
$\lambda \in L_1{}^*$



Example

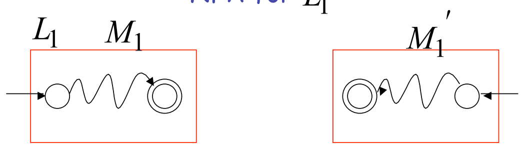
NFA for
$L_1{}^* = \{a^n b\}^*$

$w = w_1 w_2 \cdots w_k$
$w_i \in L_1$



Reverse

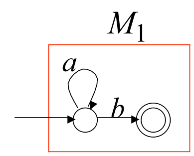
NFA for L_1^R



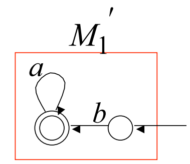
1. Reverse all transitions
2. Make initial state final state and vice versa

Example

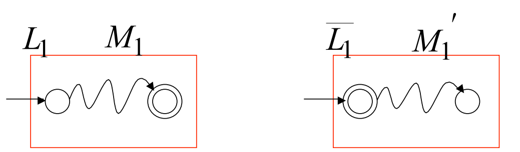
$L_1 = \{a^n b\}$



$L_1^R = \{b a^n\}$



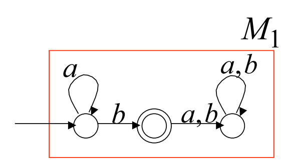
Complement



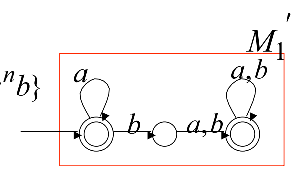
1. Take the **DFA** that accepts L_1
2. Make final states non-final, and vice-versa

Example

$L_1 = \{a^n b\}$



$\overline{L_1} = \{a,b\}^* - \{a^n b\}$



Intersection

DeMorgan's Law: $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$

L_1 , L_2 regular

$\Rightarrow$ $\overline{L_1}$, $\overline{L_2}$ regular

$\Rightarrow$ $\overline{L_1} \cup \overline{L_2}$ regular

$\Rightarrow$ $\overline{\overline{L_1} \cup \overline{L_2}}$ regular

$\Rightarrow$ $L_1 \cap L_2$ regular

What's Next

- Read
  - Linz Chapter 1,2.1, 2.2. 2.3, (skip 2.4), 3, and Chapter 4
  - JFLAP Startup, Chapter 1, 2.1, (skip 2.2), 3, 4
- Next Lecture Topics from Chapter 4.2 and 4.3
  - Properties of regular languages
  - The pumping lemma (for regular languages)
- Quiz 1 in Recitation on Wednesday 9/17
  - Covers Linz 1.1, 1.2, 2.1, 2.2, 2.3, and JFLAP 1, 2.1
  - Closed book, but you may bring one sheet of 8.5 x 11 inch paper with any notes you like.
  - Quiz will take the full hour on Wednesday
- Homework
  - Homework Due Thursday