Regular Expressions and Grammars

Formal Languages and Abstract Machines

Week 04

Baris E. Suzek, PhD

Outline

- Last week
- Regular expressions
- Grammars

Deterministic Finite Accepter (DFA)

$$M = (Q, \Sigma, \delta, q_0, F)$$

Q : set of states

Σ : input alphabet

δ : transition function is a total function <u>there must be an action defined for every combination of state and symbol</u>

q_0 : initial state

F : set of final states

Formal Definition of NFAs

$$M = (Q,\ \Sigma,\ \delta,\ q_0,\ F)$$

-

$Q:$ <u>Set of states, i.e.</u> $\{q_0, q_1, q_2\}$

$\Sigma:$ Input alphabet, i.e. $\{a, b\}$

$\delta:$ Transition function $\delta : Q \times \boxed{(\Sigma \cup \{\lambda\})} \to 2^Q.$

$q_0:$ Initial state

$F:$ Final states

NFA vs. DFA

- Transition functions range is Q vs. 2^Q (powersets of Q)
- λ can be an argument of transition function; transition without consuming a symbol
- $\delta(q_k, a)$ can be empty (not a total function)

δ	a	b
q_0	q_1	
q_1		q_2

Regular Languages

- A language L is regular if there is a DFA M such that $L = L(M)$

- All regular languages form a language family

Reverse

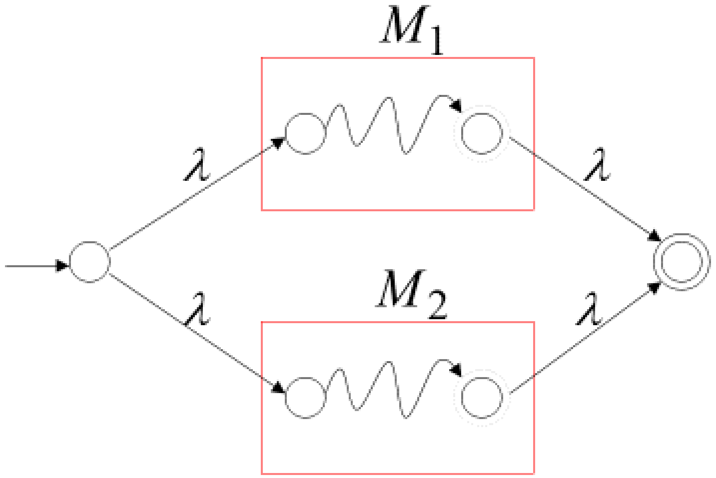
NFA for L_1^R

1. Reverse all transitions

2. Make initial state final state and vice versa

Union $L_1 \cup L_2$

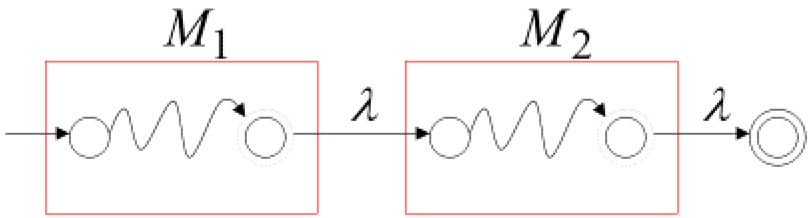
- NFA for



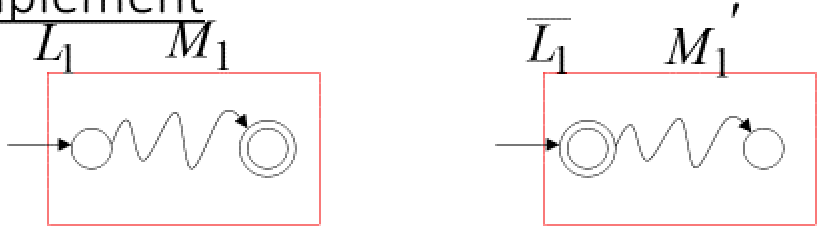
Concatenation

$L_1 L_2$

- NFA for



Complement



1. Take the **DFA** that accepts L_1

2. Make final states non-final, and vice-versa

Describing Regular Languages

- DFA or NFA (covered)
- Regular expressions
- Regular grammars

Outline

- Last week
- Regular expressions
- Grammars

Regular Expressions

- Regular expressions describe regular languages

$$(a+b\cdot c)^*$$

- Example describes the language:

$$\{a,bc\}^* = \{\lambda, a, bc, aa, abc, bca, ...\}$$

Recursive Definition

Primitive regular expressions: $\emptyset, \quad \lambda, \quad \alpha$

Given regular expressions r_1 and r_2

$$\left.\begin{array}{ll} \text{Union (or)} & r_1 + r_2 \\ \text{Concatenation} & r_1 \cdot r_2 \\ \text{Star closure} & r_1{}^{*} \\ & (r_1) \end{array}\right\} \text{Are regular expressions}$$

Examples

A regular expression: $(a+b\cdot c)^{*}\cdot(c+\varnothing)$

Not a regular expression: $(a+b+)$

Languages of Regular Expressions

$L(r)$: language of regular expression r

Example

$$L((a+b\cdot c)^*)=\{\lambda,a,bc,aa,abc,bca,...\}$$

Definition

- For primitive regular expressions:

$$L(\varnothing)=\varnothing$$

$$L(\lambda)=\{\lambda\}$$

$$L(a)=\{a\}$$

Definition (continued)

- For regular expressions r_1 and r_2
- $$L(r_1 + r_2) = L(r_1) \cup L(r_2)$$

$$L(r_1 \cdot r_2) = L(r_1)\, L(r_2)$$

$$L(r_1 *) = (L(r_1))*$$

$$L((r_1)) = L(r_1)$$

Example

- Regular expression: $(a+b)\cdot a*$

$$
\begin{aligned}
L((a+b)\cdot a*) &= L((a+b))\,L(a*) \\
&= L(a+b)\,L(a*) \\
&= (L(a)\cup L(b))\,(L(a))* \\
&= (\{a\}\cup\{b\})\,(\{a\})* \\
&= \{a,b\}\,\{\lambda,a,aa,aaa,\ldots\} \\
&= \{a,aa,aaa,\ldots,b,ba,baa,\ldots\}
\end{aligned}
$$

Example

- Regular expression: $r = (a+b)^*(a+bb)$

$$L(r) = \{a, bb, aa, abb, ba, bbb, \ldots\}$$

Example

- Regular expression: $r = (aa)^*(bb)^*b$

$$L(r) = \{a^{2n}b^{2m}b: \quad n, m \geq 0\}$$

Example

- Regular expression: $r = (0+1)^* 00 (0+1)^*$

$$L(r) = \{\text{all strings with at least two consecutive } 0\}$$

Example

- Regular expression: $r = (1 + 01)^*(0 + \lambda)$

$L(r)$ = { all strings without two consecutive 0 }

Equivalent Regular Expressions

- Regular expressions r_1 and r_2 are **equivalent** if

$$L(r_1) = L(r_2)$$

Example

- L = { all strings without two consecutive 0 }

$$r_1 = (1 + 01)^* (0 + \lambda)$$

$$r_2 = (1^* 011^*)^* (0 + \lambda) + 1^* (0 + \lambda)$$

$$L(r_1) = L(r_2) = L$$

$\Rightarrow$ r_1 and r_2 are equivalent regular expr.

Regular Expressions
and
Regular Languages

Theorem

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

Theorem - Part 1

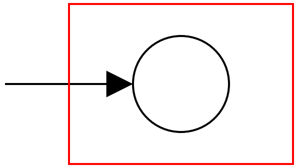
$$\left\{ \begin{array}{l} \text{Languages} \\ \text{Generated by} \\ \text{Regular Expressions} \end{array} \right\} \subseteq \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$$

1. For any regular expression r the language $L(r)$ is regular

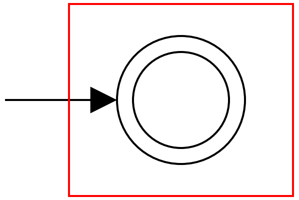
Proof - Part 1
Induction Basis

- Primitive Regular Expressions: $\emptyset,\ \lambda,\ \alpha$

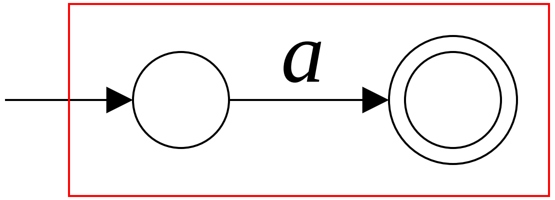
NFAs



$$L(M_1) = \emptyset = L(\emptyset)$$



$$L(M_2) = \{\lambda\} = L(\lambda)$$



$$L(M_3) = \{a\} = L(a)$$

regular languages

Proof - Part 1
Inductive Hypothesis

- Assume for regular expressions r_1 and r_2 that $L(r_1)$ and $L(r_2)$ are regular languages

Proof - Part 1
Inductive Step

- We will prove:

$$\left.\begin{array}{l} L(r_1 + r_2) \\ \\ L(r_1 \cdot r_2) \\ \\ L(r_1 *) \\ \\ L((r_1)) \end{array}\right\} \text{Are regular Languages}$$

- By definition of regular expressions:

$$L(r_1 + r_2) = L(r_1) \cup L(r_2)$$

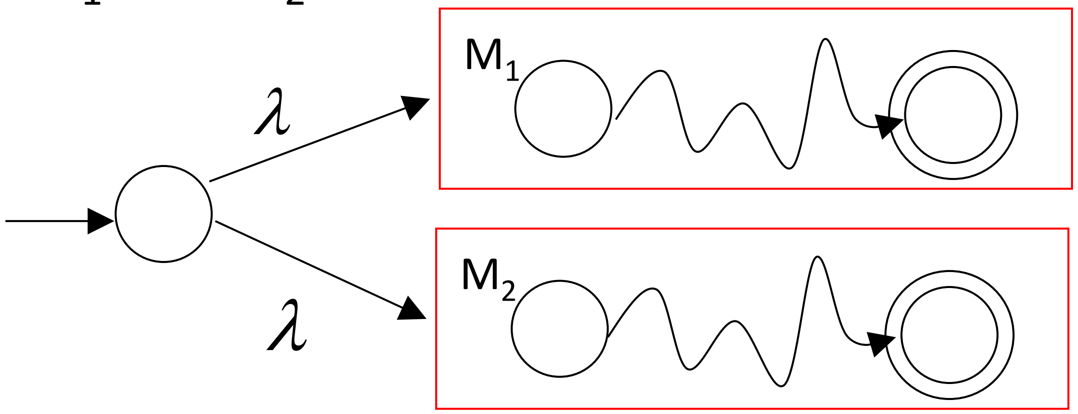
$$L(r_1 \cdot r_2) = L(r_1)\, L(r_2)$$

$$L(r_1 *) = (L(r_1))*$$

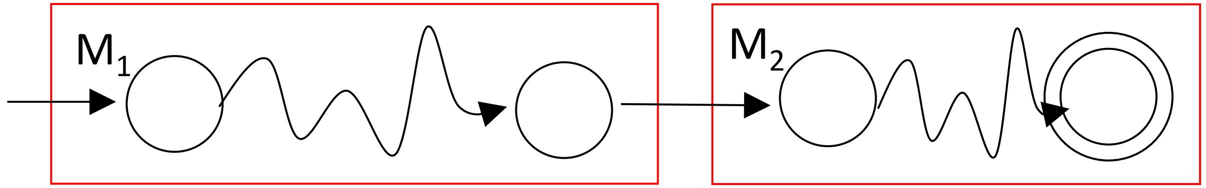
$$L((r_1)) = L(r_1)$$

By inductive hypothesis we know are $L(r_1)$ and $L(r_2)$ regular languages. There exists single final state NFAs M_1 and M_2 that accepts them so:

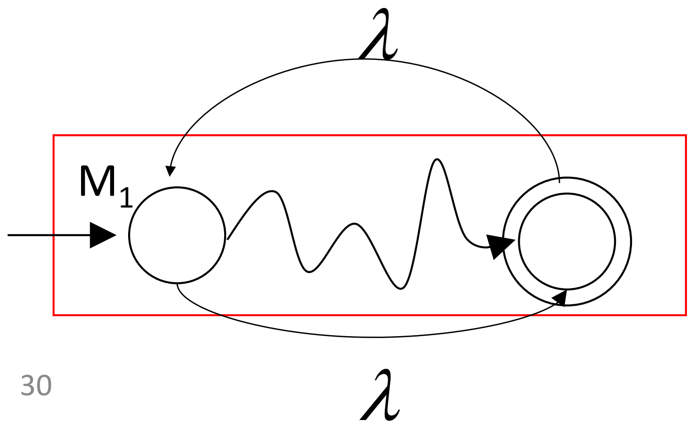
$$L(r_1 + r_2) = L(r_1) \cup L(r_2)$$



$$L(r_1 \cdot r_2) = L(r_1)\, L(r_2)$$



$$L(r_1 *) = (L(r_1))*$$



- Therefore:

$$\left.\begin{aligned}
&L(r_1 + r_2) = L(r_1) \cup L(r_2) \\[2em]
&L(r_1 \cdot r_2) = L(r_1)\, L(r_2) \\[2em]
&L(r_1 *) = (L(r_1))*
\end{aligned}\right\}\ \text{Are regular languages}$$

Theorem - Part 2

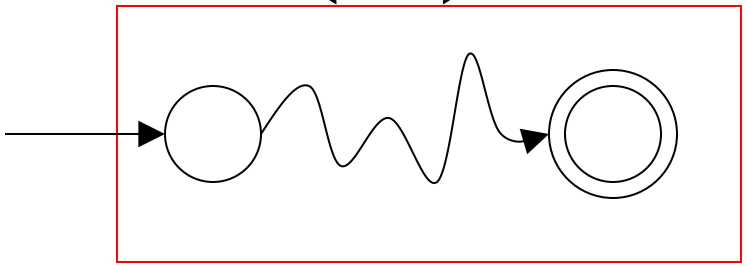
$$\left\{ \begin{array}{l} \text{Languages} \\ \text{Generated by} \\ \text{Regular Expressions} \end{array} \right\} \supseteq \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$$

2. For any regular language L there is a regular expression r with $L(r) = L$

Proof – Part 2

- Since L is regular take the NFA M that accepts it

$$L(M) = L$$

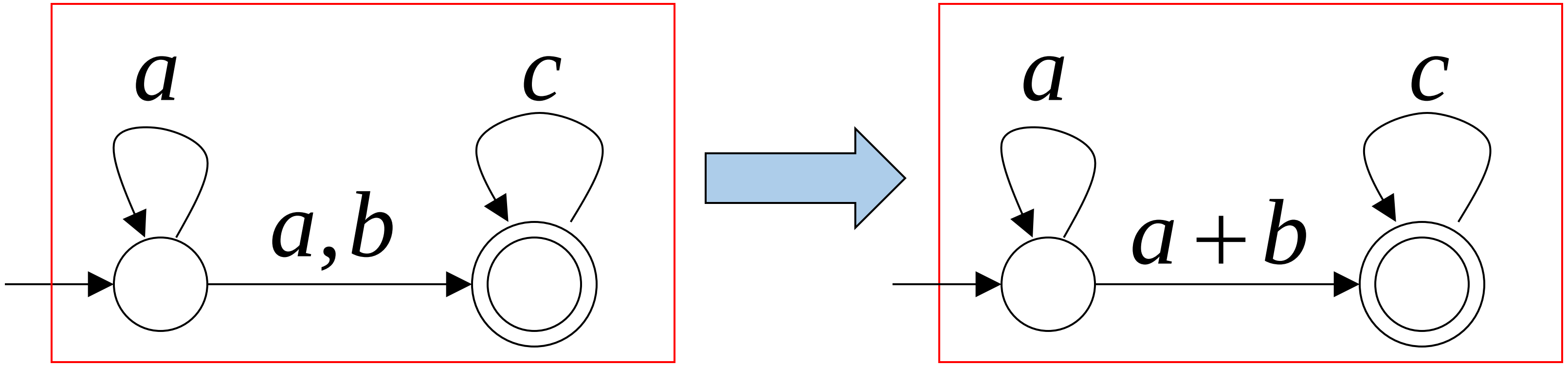


Single final state

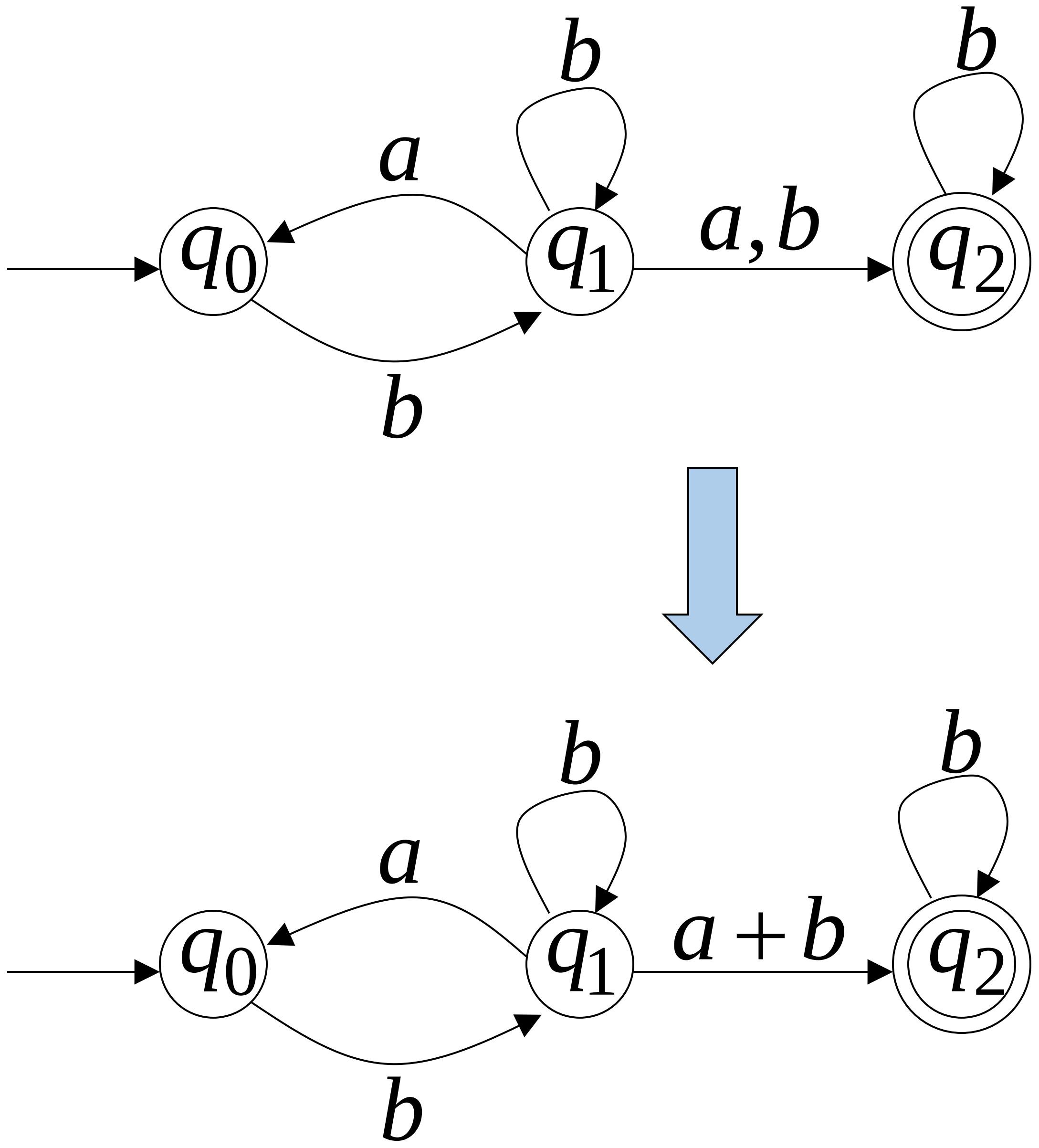
- From M construct the equivalent **Generalized Transition Graph** in which transition labels are regular expressions

Example:

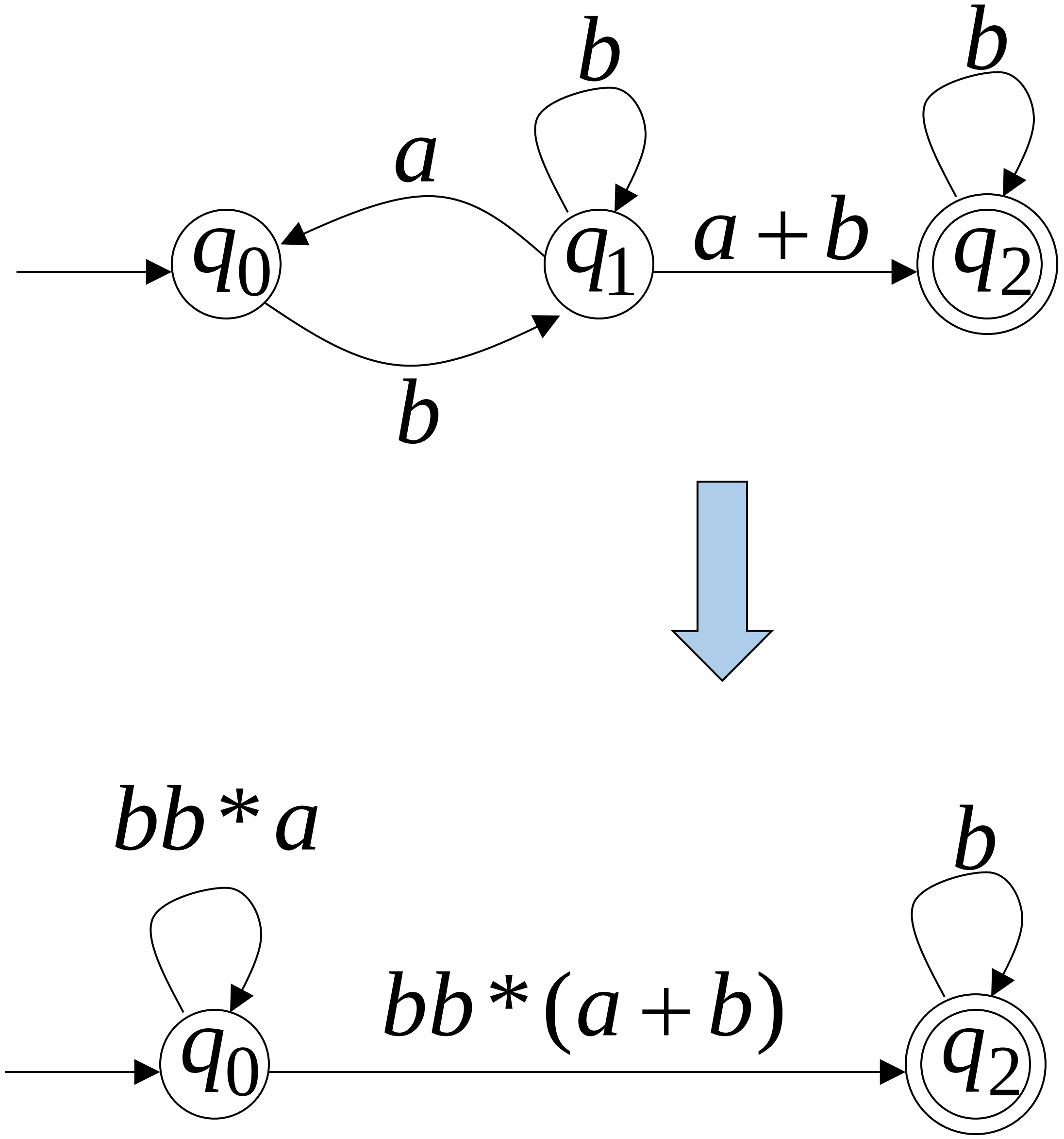
M



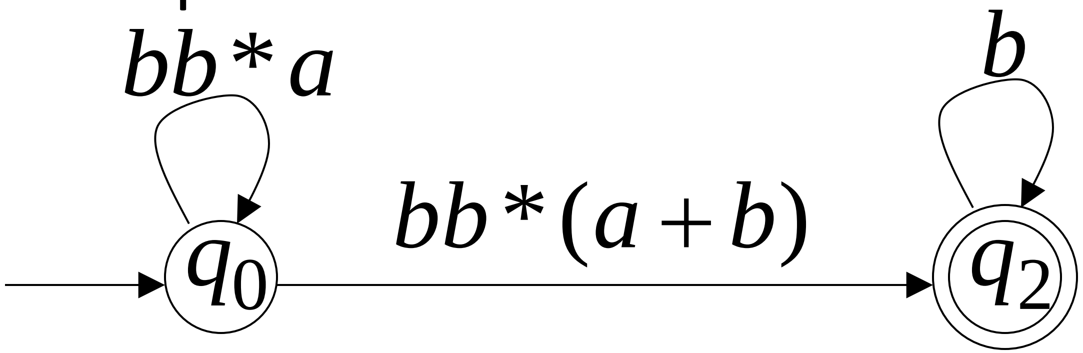
- Another Example:



- Reducing the states:



- Resulting Regular Expression:

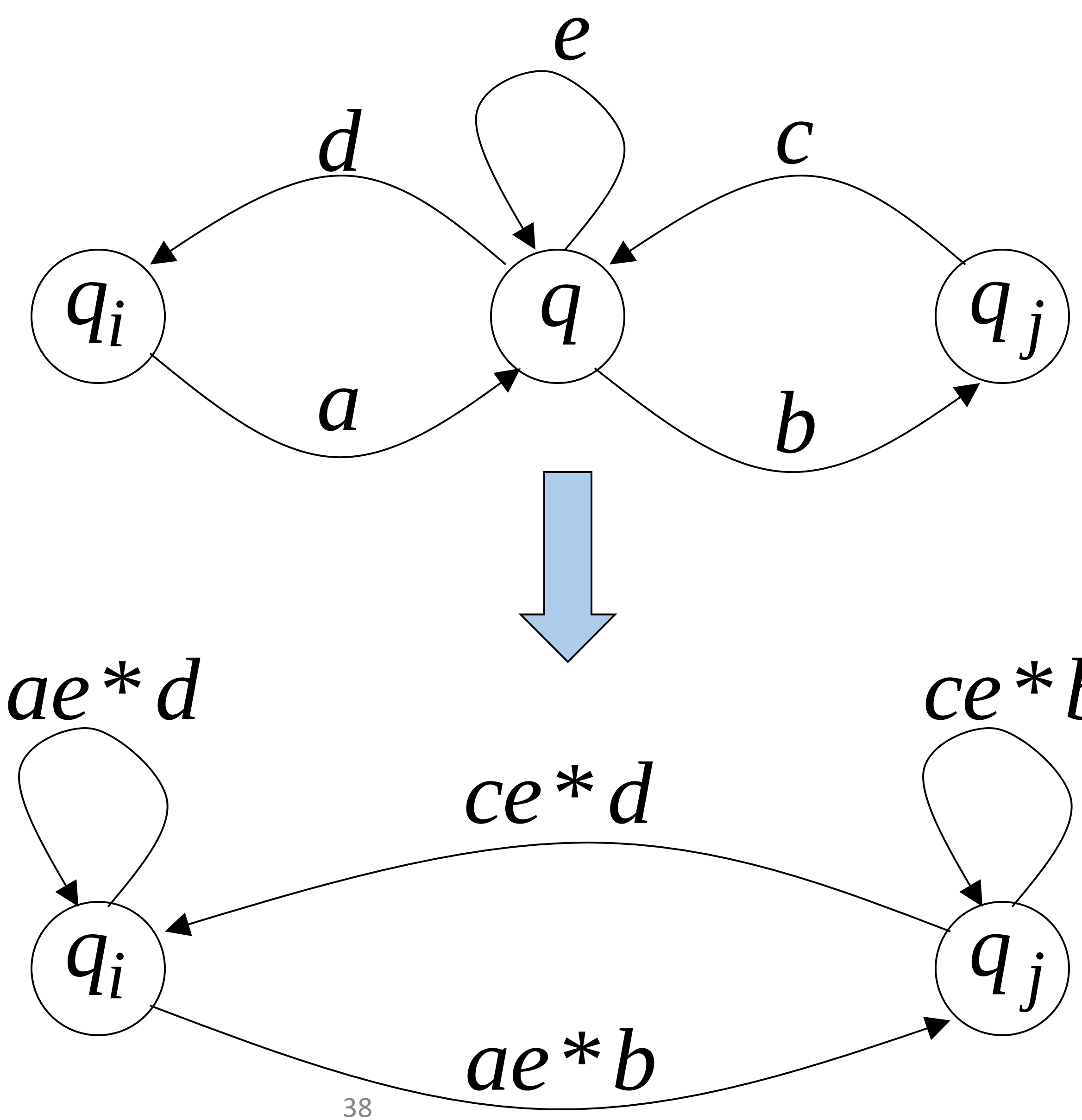


$$r = (bb^*a)^*bb^*(a+b)b^*$$

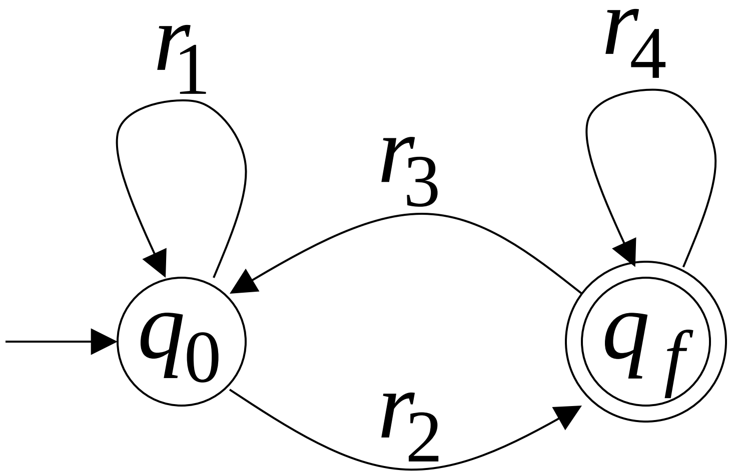
$$L(r) = L(M) = L$$

In General

- Removing states:



* The final transition graph:

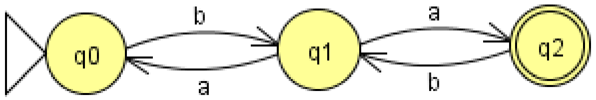


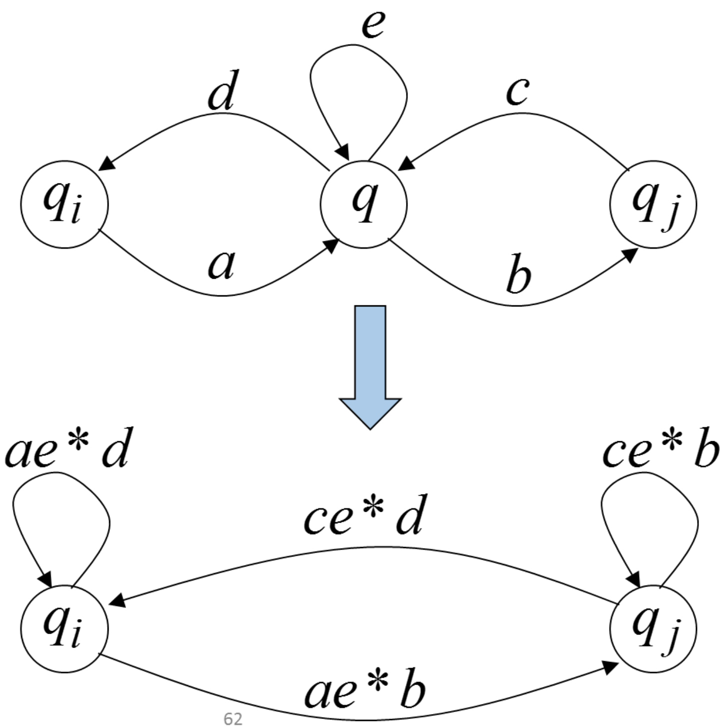
The resulting regular expression:

$$r = r_1^* r_2 (r_4 + r_3 r_1^* r_2)^*$$

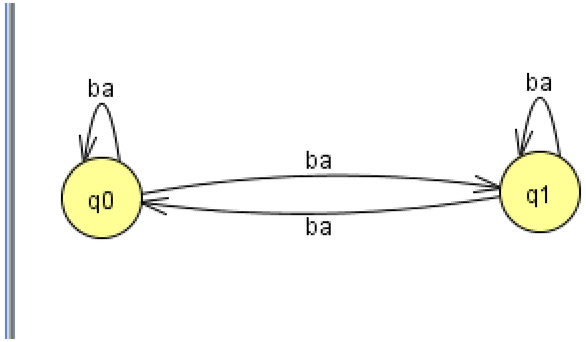
$$L(r) = L(M) = L$$

An Exercise

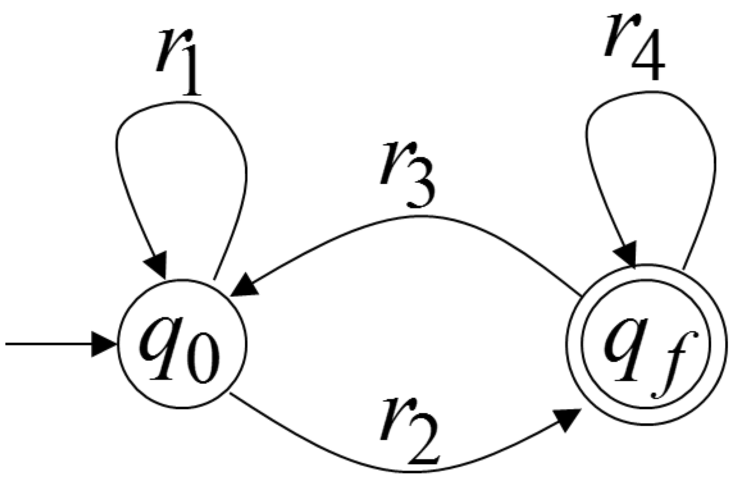




a	b
b	a
c	b
d	a
e	{lamda}

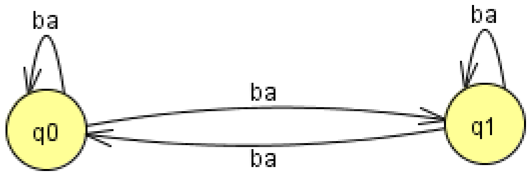


An Exercise



r1	ba
r2	ba
r3	ba
r4	ba

$$r = r_1{}^* r_2 (r_4 + r_3 r_1{}^* r_2)^*$$



(ba)*(ba)((ba)+(ba)(ba)*(ba))*

OR ((ba)*ba(ba)*ba)*(ba)*ba(ba)*

Some Linux and Regular Expressions

- more
- gz
- cat
- od
- cut
- join
- sort
- paste
- grep
- awk/sed
- shuf
- Wc
- head{tail

Outline

- Last week
- Regular expressions
- Grammars

Describing Regular Languages

- DFA or NFA (covered)
- Regular expressions (covered)
- Regular grammars

Grammars

- Grammars express languages

- Example: the English language

$$\langle sentence \rangle \to \langle noun\_phrase \rangle \ \langle verb \rangle$$

$$\langle noun\_phrase \rangle \to \langle article \rangle \ \langle noun \rangle$$

$$\langle article \rangle \to a$$

$$\langle article \rangle \to the$$

$$\langle noun \rangle \to cat$$

$$\langle noun \rangle \to dog$$

$$\langle verb \rangle \to runs$$

$$\langle verb \rangle \to walks$$

- A derivation of "the dog walks":

$$
\begin{aligned}
\langle sentence \rangle &\Rightarrow \langle noun\_phrase \rangle \ \langle verb \rangle \\
&\Rightarrow \langle article \rangle \ \langle noun \rangle \ \langle verb \rangle \\
&\Rightarrow the \ \ \langle noun \rangle \ \langle verb \rangle \\
&\Rightarrow the \ \ dog \ \ \langle verb \rangle \\
&\Rightarrow the \ \ dog \ \ walks
\end{aligned}
$$

- A derivation of “a cat runs”:

$$
\begin{aligned}
\langle sentence \rangle &\Rightarrow \langle noun\_phrase \rangle \ \langle verb \rangle \\
&\Rightarrow \langle article \rangle \ \langle noun \rangle \ \langle verb \rangle \\
&\Rightarrow a \ \ \langle noun \rangle \ \langle verb \rangle \\
&\Rightarrow a \ \ cat \ \ \langle verb \rangle \\
&\Rightarrow a \ \ cat \ \ runs
\end{aligned}
$$

Language of the Grammar

$$\langle article \rangle \to a$$
$$\langle article \rangle \to the$$

$$\langle noun \rangle \to cat$$
$$\langle noun \rangle \to dog$$

$$\langle verb \rangle \to runs$$
$$\langle verb \rangle \to walks$$

+

$$\langle sentence \rangle \to \langle noun\_phrase \rangle \ \langle verb \rangle$$

$$\langle noun\_phrase \rangle \to \langle article \rangle \ \langle noun \rangle$$

⇒

L = { "a cat runs",
"a cat walks",
"the cat runs",
"the cat walks",
"a dog runs",
"a dog walks",
"the dog runs",
"the dog walks" }

Notation

Production Rules

$$\underbrace{\begin{array}{l}\langle noun \rangle \rightarrow cat \\ \langle noun \rangle \rightarrow dog\end{array}}$$

Variable

Terminal

Another Example

- Grammar:

$$S \to aSb$$
$$S \to \lambda$$

- Derivation of sentence : ab

$$S \Rightarrow aSb \Rightarrow ab$$

$$S \to aSb \qquad\qquad S \to \lambda$$

Another Example

- Grammar: $S \to aSb$

  $S \to \lambda$

- Derivation of sentence : $aabb$

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$$

$S \to aSb$ $\qquad$ $S \to \lambda$

Another Example

- Language of the grammar

$$S \to aSb$$

$$S \to \lambda$$

$$L = \{a^n b^n : n \geq 0\}$$

- This is not a “regular language”
  - No DFA can accept this
  - We will learn one more method to test regular-ness: “Pumping Lemma”

More Notation

- <u>Grammar:</u>

$$G = (V, T, S, P)$$

$V:$ Set of variables

$T:$ Set of terminal symbols

$S:$ Start variable

$P:$ Set of production rules

Example

$$G \qquad S \rightarrow aSb$$

$$S \rightarrow \lambda$$

$$G = (V, T, S, P)$$

$$V = \{S\} \qquad T = \{a, b\}$$

$$P = \{S \rightarrow aSb,\ \ S \rightarrow \lambda\}$$

More Notation

$$S \to aSb$$
$$S \to \lambda$$

$$\Longrightarrow \quad S \to aSb \mid \lambda$$

$$\langle article \rangle \to a$$
$$\langle article \rangle \to the$$

$$\Longrightarrow \quad \langle article \rangle \to a \mid the$$

More Notation

- Sentential Form: A sentence that contains both variables and terminals

- Example:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$$

Sentential Forms

sentence

More Notation

- In general we write (similar to extended transition function):

$$w_1 \overset{*}{\Rightarrow} w_n$$

- If: $$w_1 \Rightarrow w_2 \Rightarrow w_3 \Rightarrow \cdots \Rightarrow w_n$$

- Note: $$w \overset{*}{\Rightarrow} w$$

Example

- We write:

$$S \overset{*}{\Rightarrow} aaabbb$$

- Instead of:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$$

Example

Grammar

$$S \rightarrow aSb$$

$$S \rightarrow \lambda$$

Derivations

$$S \overset{*}{\Rightarrow} \lambda$$

$$S \overset{*}{\Rightarrow} ab$$

$$S \overset{*}{\Rightarrow} aabb$$

$$S \overset{*}{\Rightarrow} aaabbb$$

Another Grammar Example

- Grammar G :

$$S \to Ab$$

$$A \to aAb$$

$$A \to \lambda$$

- Derivations:

$$S \Rightarrow Ab \Rightarrow b$$

$$S \Rightarrow Ab \Rightarrow aAbb \Rightarrow abb$$

$$S \Rightarrow Ab \Rightarrow aAbb \Rightarrow aaAbbb \Rightarrow aabbb$$

More Derivations

$$S \to Ab$$
$$A \to aAb$$
$$A \to \lambda$$

$$S \Rightarrow Ab \Rightarrow aAbb \Rightarrow aaAbbb \Rightarrow aaaAbbbb$$
$$\Rightarrow aaaaAbbbbb \Rightarrow aaaabbbbb$$

$$S \overset{*}{\Rightarrow} aaaabbbbb$$

$$S \overset{*}{\Rightarrow} aaaaaabbbbbbb$$

$$S \overset{*}{\Rightarrow} a^n b^n b$$

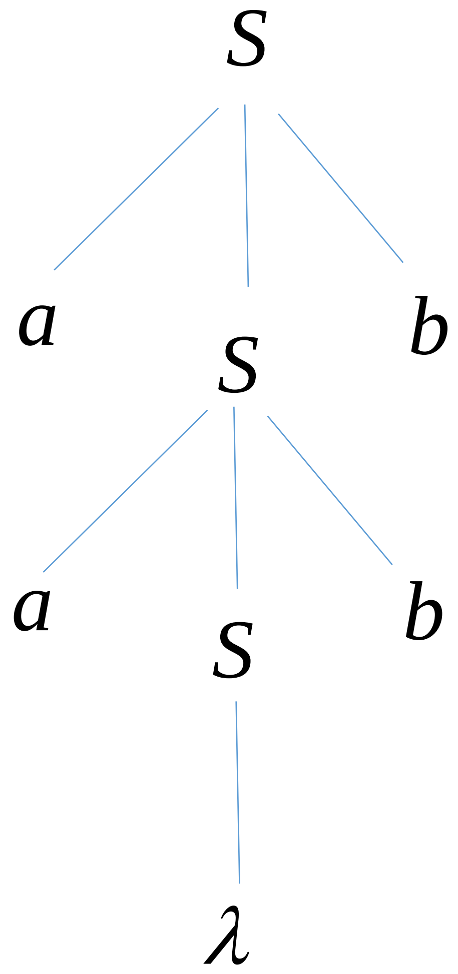
More Notation

- Parse trees: Another representation for derivations where:
    - Each interior node is a variable
    - Each leaf is a variable or terminal or λ
        - If λ then no more child

Example

$$S \to aSb$$

$$S \to \lambda$$



$$S \overset{*}{\Rightarrow} aabb$$

Language of a Grammar

- For a grammar G with start variable S :

$$L(G) = \{w : \quad S \overset{*}{\Rightarrow} w\}$$

String of terminals

Example

- For grammar G :

$$S \to Ab$$

$$A \to aAb$$

$$A \to \lambda$$

$$L(G) = \{a^n b^n b : \quad n \geq 0\}$$

Since: $\quad S \overset{*}{\Rightarrow} a^n b^n b$