CS 3186
Properties of Regular Languages
Section 4.1

Introduction

We have seen Regular Languages expressed by NFA/DFA, Regular Expression, Regular Grammar.

For a regular language, there exists an NFA that accepts the same language, that is expressed by a regular expression or a regular grammar.

(Note that in Chapter 3, we indicated that how to convert an NFA with many final states to an equivalent NFA with a single final state by a new final state and adding lambda transitions to the new final state. It is easier to deal with an NFA with one initial and one final state.)

Operations and Closure

Closure of a set with respect to an operator: An operation is performed on operands from a set and the output result also belongs to the same set.

The set is then said to be closed under this operation. i.e., there is no new element that is added to the set because of this operation.

Consider set of natural numbers N = {0, 1, 2, 3, …}
Consider arithmetical operators (+, -, x, /) that typically operate on input operands resulting in an output.
(Binary operations with two operands: 3+2=5   2-3=-1 3x2=6   3/2=1.5
Unary operator "-" for negation: -2
Set N is closed under + and x as the output always belongs to N
Set N is not closed under – and /. (-1, -2, and 1.5 does not belong to N)

Consider set of Integers I = {…-3, -2, -1, 0, 1, 2, 3,…}
Consider arithmetical operators (+, -, x, /)
Set I is closed under +, -, and x and is not closed under /.

Closure of Regular Languages under Set Operations

For regular languages L1 and L2, we will prove that:

Union: $L_1 \cup L_2$

Concatenation: $L_1 L_2$

Star: $L_1{}^*$

Reversal: $L_1{}^R$

Complement: $\overline{L_1}$

Intersection: $L_1 \cap L_2$

Difference: $L_1 - L_2$

Are also Regular Languages

Regular languages are **closed** under the following set operations.

Union: $L_1 \cup L_2$

Concatenation: $L_1 L_2$

Star: $L_1{}^*$

Reversal: $L_1{}^R$

Complement: $\overline{L_1}$

Intersection: $L_1 \cap L_2$

Difference: $L_1 - L_2$

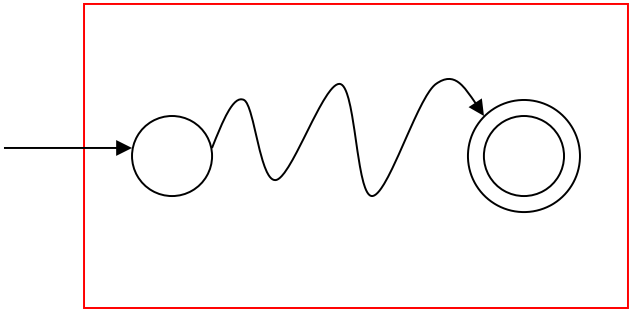
Notation

Regular language L_1

There exists NFA M_1

$$L(M_1) = L_1$$

NFA M_1



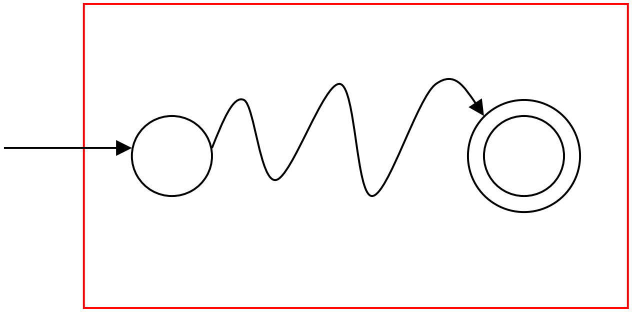
There exists a regular expression r_1 such that $L(r_1)=L_1$ (See Chapter 3)

Regular language L_2

There exists NFA M_2

$$L(M_2) = L_2$$

NFA M_2



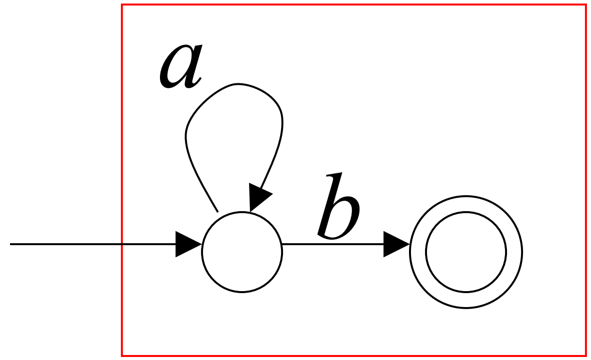
There exists a regular expression r_2 such that $L(r_2)=L_2$

Example

M_1

$n \geq 0$

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

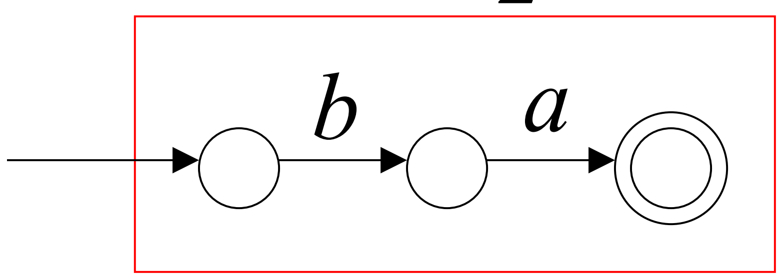


r₁ = a*b

L(r₁)= L₁

M_2

$L_2 = \{ba\}$

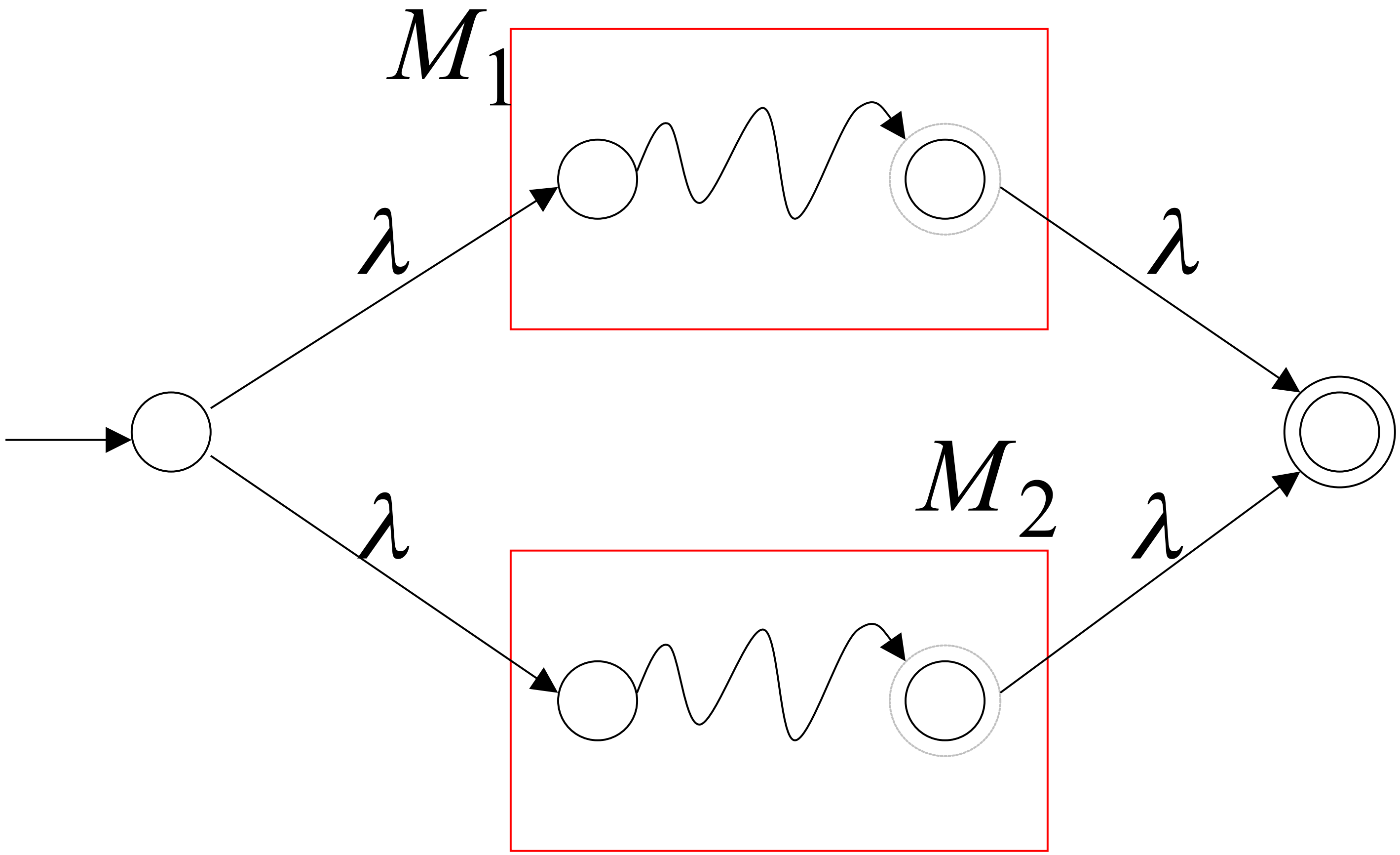


r₂ = ba

L(r₂) = L₂

Union

We can construct NFA for $L_1 \cup L_2$ as follows:



If w is accepted by M_1 or M_2, then w is accepted by the above construction. Hence, the above NFA accepts $L_1 \cup L_2$

Hence a Regular Language is closed under Union operator.

Union (Another proof)

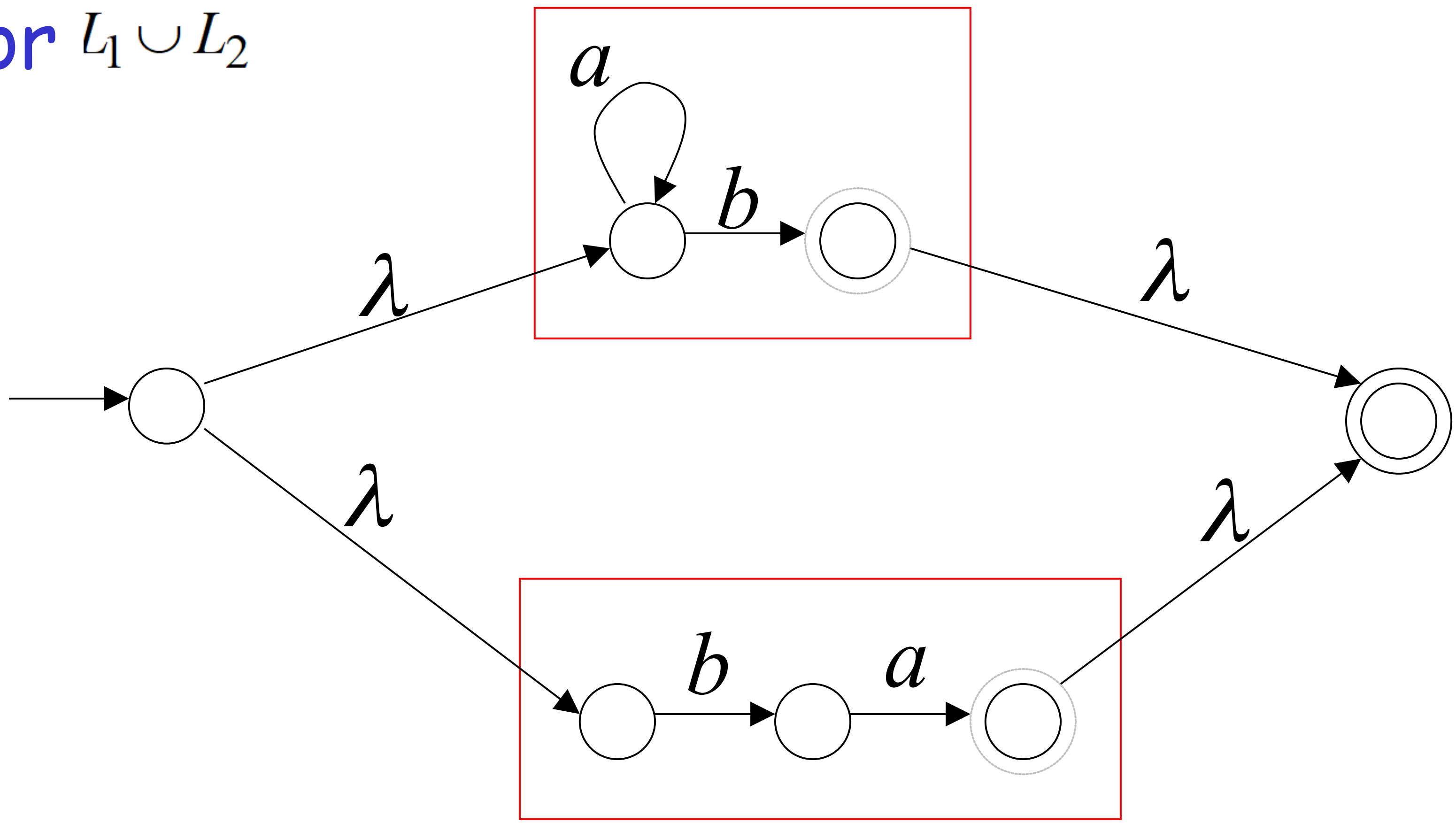
If r_1 is a regular expression such that $L(r_1)= L_1$ and
If r_2 is a regular expression such that $L(r_2)= L_2$

We can describe a regular expression for $L_1 \cup L_2$ as simply $r_1 + r_2$. (See Chapter 3.1)

Hence a Regular Language is closed under Union operator.

Example

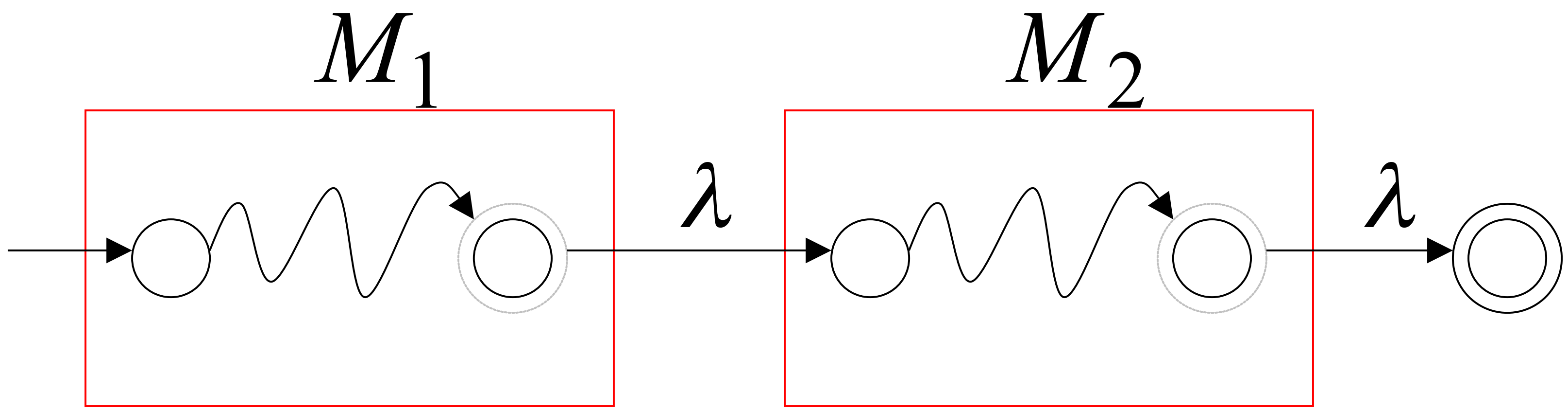
NFA for $L_1 \cup L_2$



Regular expression for $L_1 \cup L_2$ =r1 + r2

Concatenation

We can construct NFA for $L_1 L_2$ as follows:



If x is accepted by M_1 and y is accepted by M_2, then xy is accepted by the above construction.

Hence a Regular Language is closed under Concatenation operator.

Concatenation (Another proof)

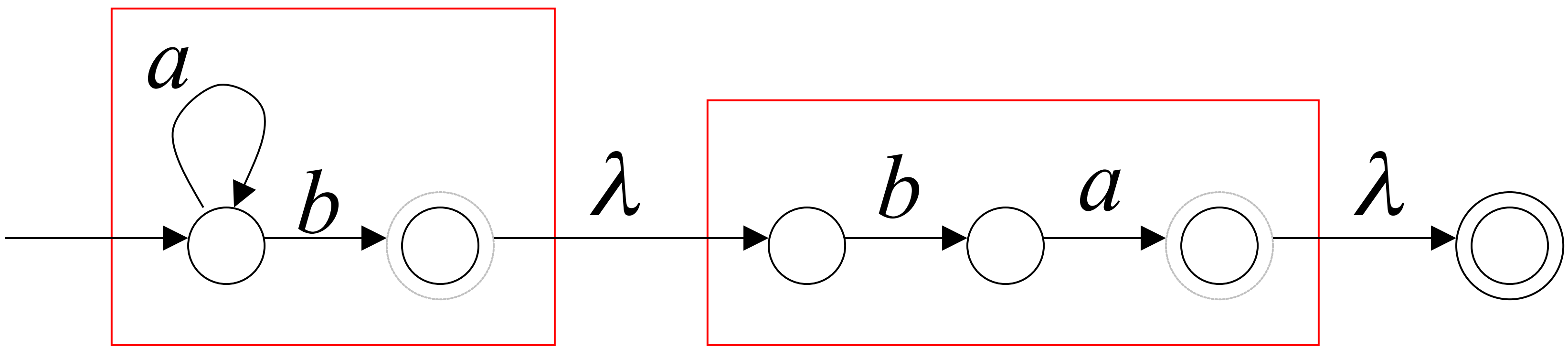
If r_1 is a regular expression such that $L(r_1) = L_1$ and
If r_2 is a regular expression such that $L(r_2) = L_2$

We can describe a regular expression for $L_1 L_2$ as simply $r_1 r_2$. (See Chapter 3.1)

Hence a Regular Language is closed under Concatenation operator.

Example

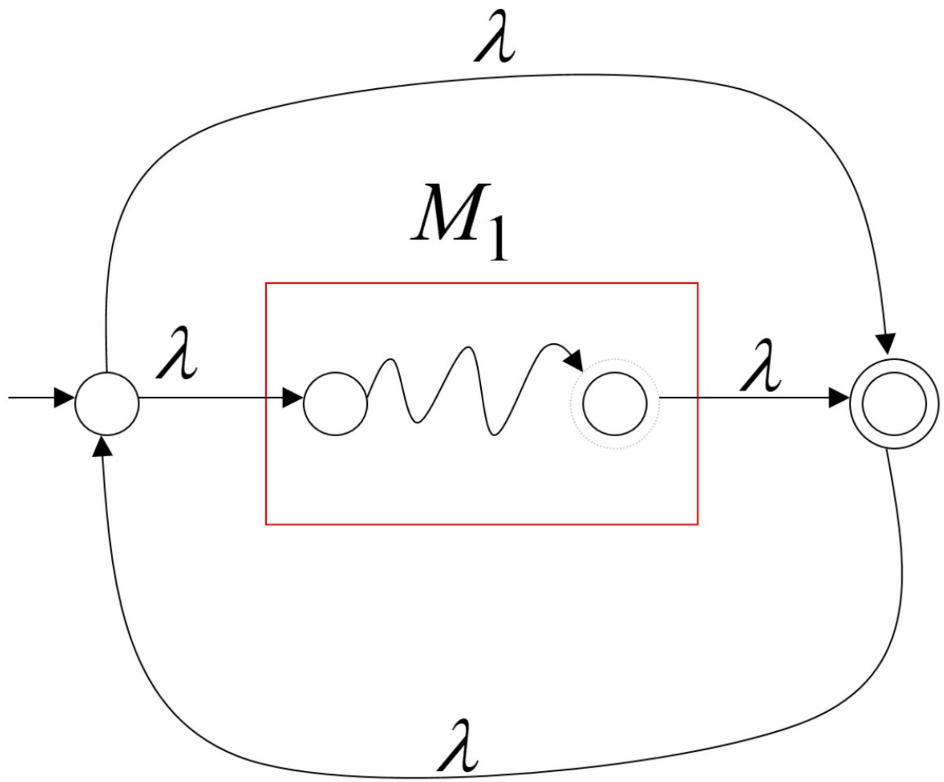
NFA for $L_1 L_2$



Regular expression for $L_1 L_2 = r_1\, r_2$

Star Operation

We can construct NFA for $L_1 *$ as follows:



Any string in w is accepted by the NFA. $w = w_1 w_2 \cdots w_k$

$w_i \in L_1$

Hence a Regular Language is closed under Star operator.

Star (Another proof)

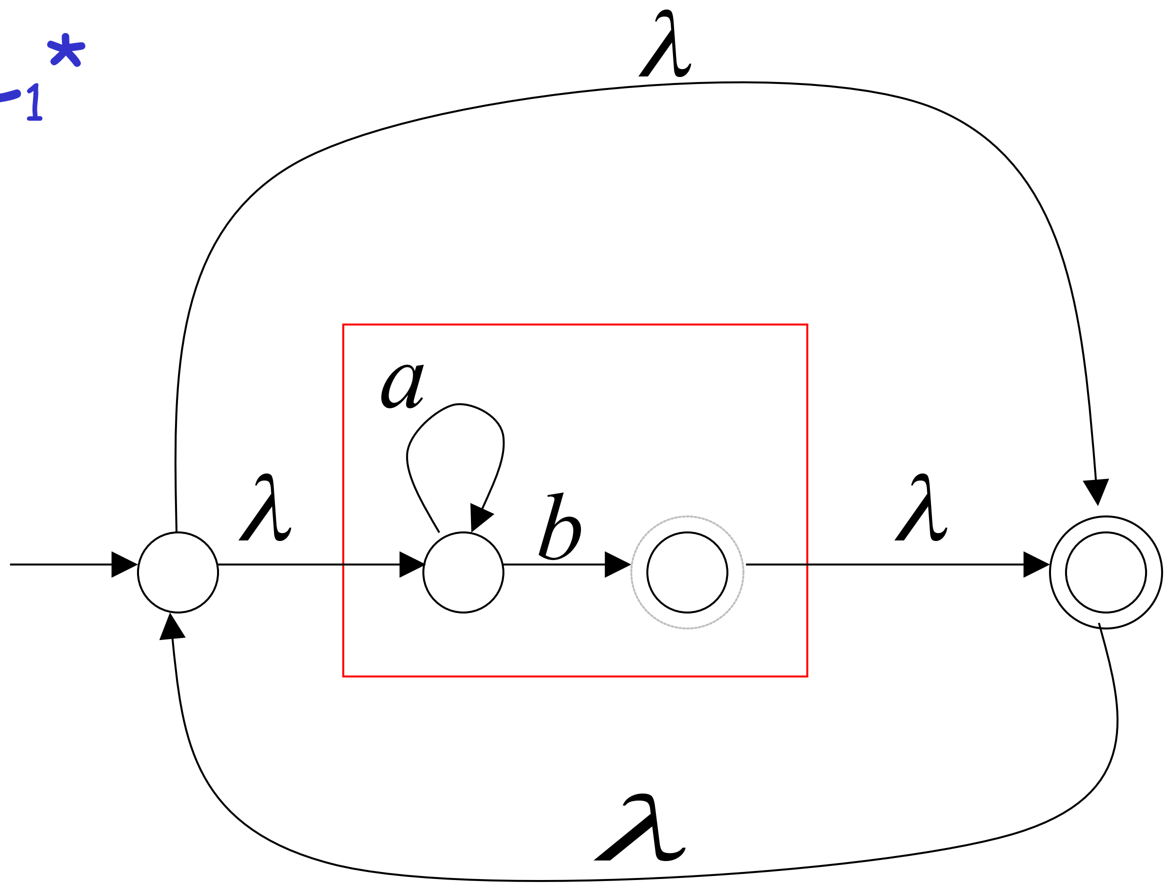
If r_1 is a regular expression such that $L(r_1)= L_1$

We can describe a regular expression for $L_1{}^*$ as simply $r_1{}^*$ (See Chapter 3.1)

Hence a Regular Language is closed under Star operator.

Example

NFA for L_1*



Regular expression for $L_1^* = (a^*b)^*$

Reverse

We can construct NFA for L_1^R as follows:

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

If w is accepted by M_1, w^R is accepted by M_1'

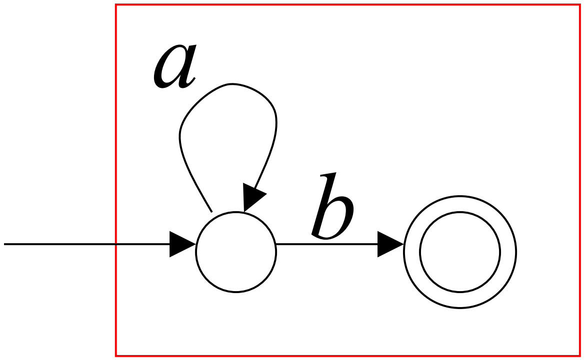
Hence M_1' accepts L_1^R

Hence a Regular Language is closed under Reverse operator.

Example

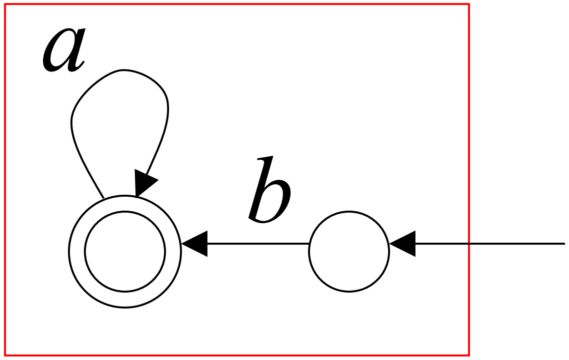
$$L_1 = \{a^n b\}$$
$$n \geq 0$$

M_1



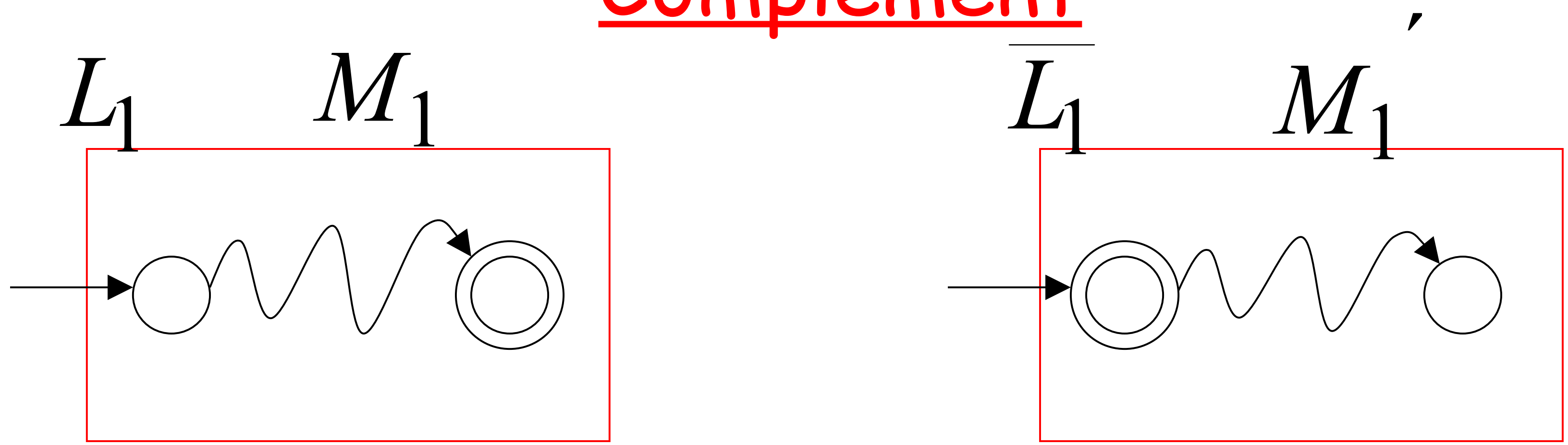
$$L_1^R = \{b a^n\}$$
$$n \geq 0$$

M_1'



(M_1' may become an NFA even if M_1 is a DFA)

Complement



We can construct DFA for $\overline{L_1}$ as follows:

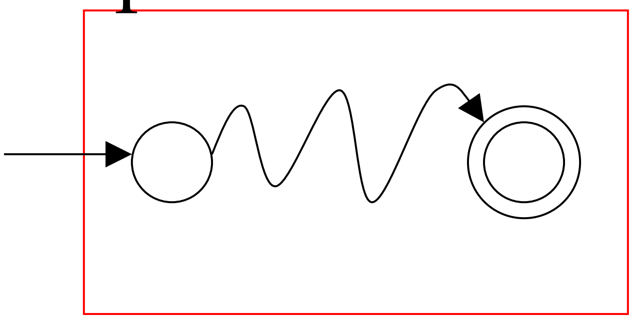
1. Take the **DFA**, **M_1**, that accepts L_1
2. Make final states non-final, and vice-versa.
3. The initial state remains the same.

Then if w is accepted by M_1, w is not accepted by M_1' and vice-versa. i.e., all the strings that are not accepted by M_1 are accepted by M_1' Hence, M_1' accepts $\overline{L_1}$

Hence a Regular Language is closed under Complement operator

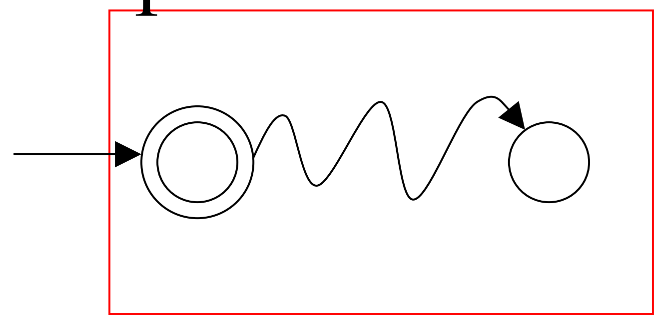
Complement (continued)

L_1



$M_1 = (Q, \Sigma, \delta, q_0, F)$

$\overline{L_1}$



$M_1' = (Q, \Sigma, \delta, q_0, Q - F)$

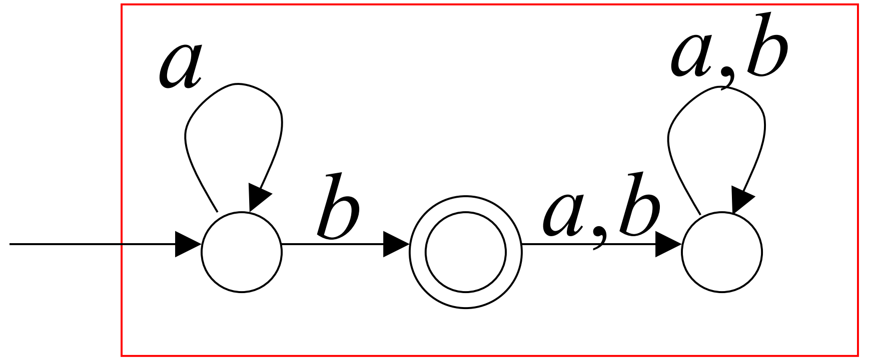
Note 1: This process works only on a DFA and does not work on an NFA. If we have NFA rather than DFA, we must first convert it to DFA and apply this construction to get its complement.

Note 2: Since a language is regular if and only if it is accepted by some NFA, **the complement of a regular language is also regular**.

Example

M_1

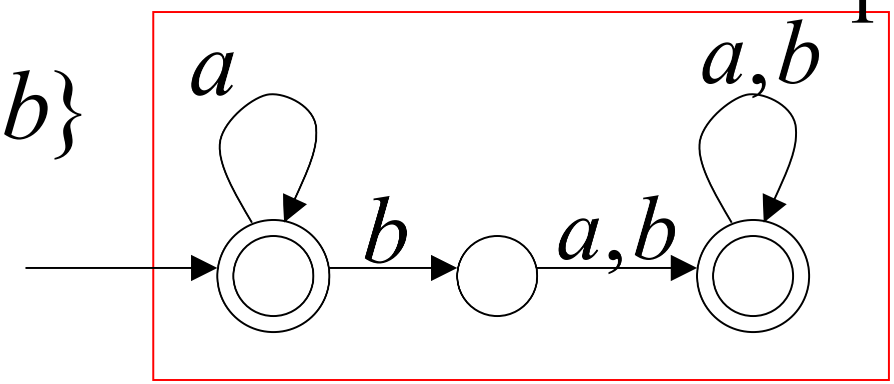
$$L_1 = \{a^n b\}$$
$$n \geq 0$$



,

M_1

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



Note: L_1 and $\overline{L_1}$ are described in set notation and not as a regular expression.

The regular expression for L_1 = a*b

Can the regular expression for L_1 be described by (a+b)* - a*b?

(No! The "-" operator is not defined for regular expression)

Intersection

We can construct NFA for $L_1 \cap L_2$ as follows:

Let $L_1 = L(M_1)$ and $L_2 = L(M_2)$ for dfas:

$$M_1 = (Q, \Sigma, \delta_1, q_0, F_1)$$

$$M_2 = (P, \Sigma, \delta_2, p_0, F_2)$$

Construct $\widehat{M} = (\widehat{Q}, \Sigma, \widehat{\delta}, (q_0, p_0), \widehat{F})$, where

$$\widehat{Q} = Q \times P$$

$$\widehat{\delta}((q_i, p_j), a) = (q_k, p_l) \textbf{ when } \delta_1(q_i, a) = q_k$$

$$\delta_2(p_j, a) = p_l$$

$$\widehat{F} = \{(q, p) : q \in F_1, p \in F_2\}$$

Clearly, $\omega \in L_1 \cap L_2$ if and only if ω accepted by $\widehat{M}$.

Thus, $L_1 \cap L_2$ is regular.

Hence a Regular Language is closed under Intersection operator.

Intersection (Another proof)

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

Difference

$L_1 - L_2$ contains all strings from L_1 but not in L_2

All strings not in L_2 = all strings in complement of L_2

i.e., $L_1 - L_2$ contains strings that are in L_1 and complement of L_2

i.e., $$L_1 - L_2 = L_1 \cap \overline{L_2}$$

Sinc

L_1 , L_2 regular

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

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

$\Rightarrow$ $L_1 - L_2$ regular

An NFA c

Hence a Regular Language is closed und rence operator.

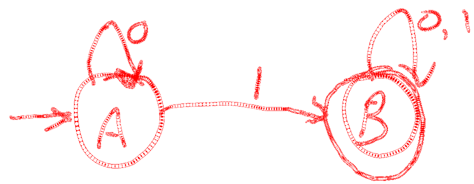
$L_1 \cap \overline{L_2}$

Complete Example

$\Sigma = \{0,1\}$

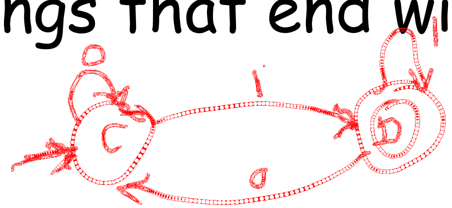
(i) Give a DFA, M_1, that accepts a Language

L_1 = {all strings that contains a 1; i.e., at least one 1}



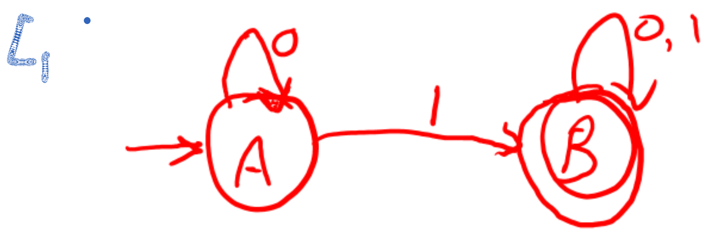
(ii) Give a DFA, M_2, that accepts a Language

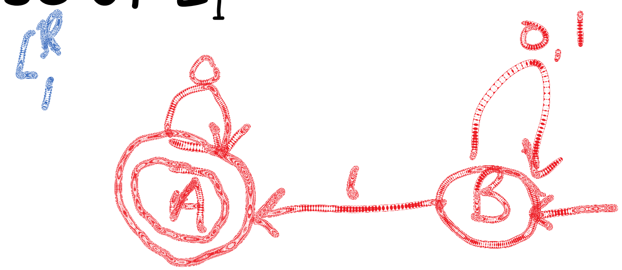
L_2 = {all strings that end with 1}



Example (continued)

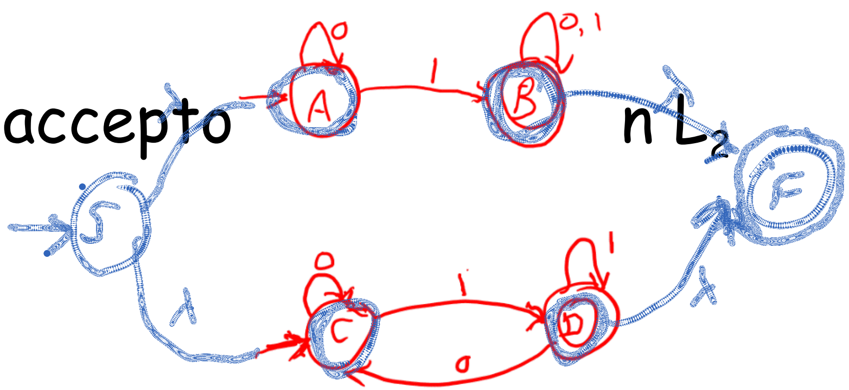
(iii) Give acceptor for Reverse of L_1





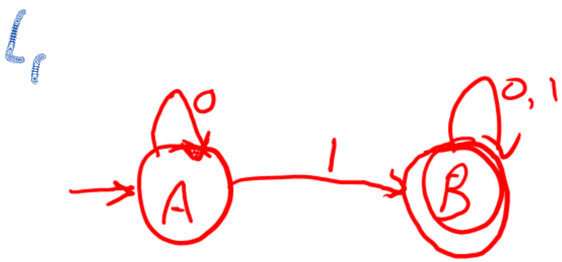
(iv) Give acceptor for complement of L_2

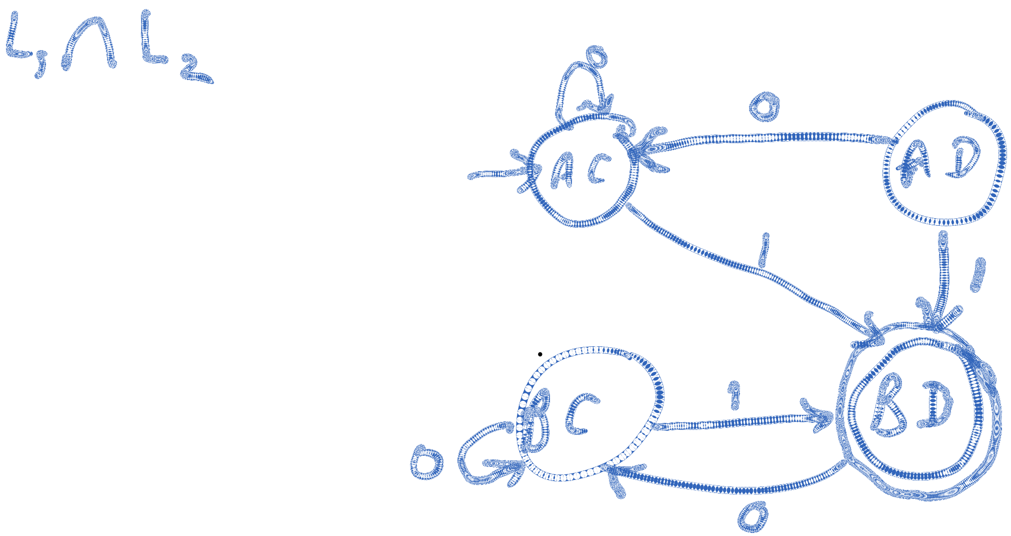
(v) Give accepto n L_2



Example (continued)

(vi) Give acceptor for L_1 intersection L_2



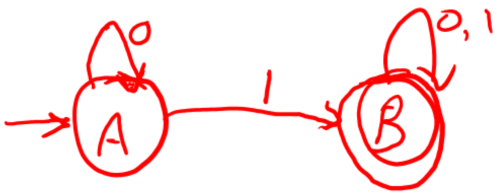


Example (continued)

(vii) Give acceptor for $L_1 - L_2$

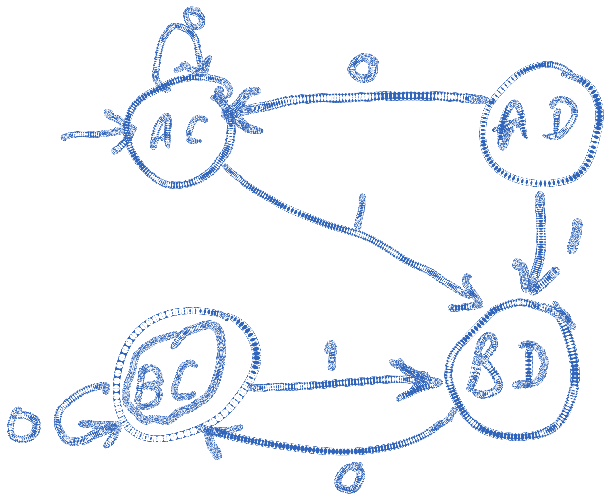
$L_1 - L_2 = L_1 \cap \bar{L}_2$

L_1



$\bar{L}_2$

$L_1 \cap \bar{L}_2$



Homework 4.1

$\Sigma = \{0,1\}$

(I) Give a DFA that accepts language L with all strings over that begins with a 0.

(i) What is the regular expression of L

(ii) Give a DFA that accepts L^R

(iii) Give a DFA that accepts $\overline{L}$

(II) (i) Give a DFA, M_1, that accepts a Language L_1 that contains even number of 0's. (Hint: only 2 states)

(ii) Give a DFA, M_2, that accepts a Language L_2 that contains even number of 1's.

(iii) Give acceptor for Reverse of L_1

(iv) Give acceptor for complement of L_2

(v) Give acceptor for L_1 union L_2

(vi) Give acceptor for L_1 intersection L_2

(vii) Give acceptor for $L - L$

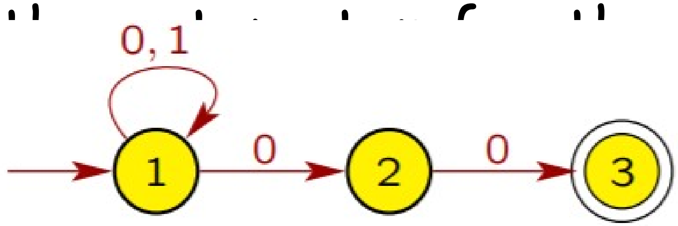
Homework 4.1

(III) Given $L_1 = \{a^n b \mid n > 0\}$ $L_2 = \{ab, ba\}$ $\Sigma = \{a, b\}$

(i) Give acceptor for L_1 intersection L_2

(ii) Give acceptor for $L_1 - L_2$

(IV) Given



(ii) Give an automaton that accepts L^R

(iii) Give a automata that accepts $\overline{L}$

(V) Homework 2.1 exercises considered many languages. Make up problems similar to the problems described in #II.