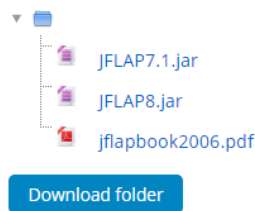


- (1) Download JFLAP from the folder: “JFLAP: graphical tools for Formal Languages and Automata Theory” at the top of the Labs page on Moodle.

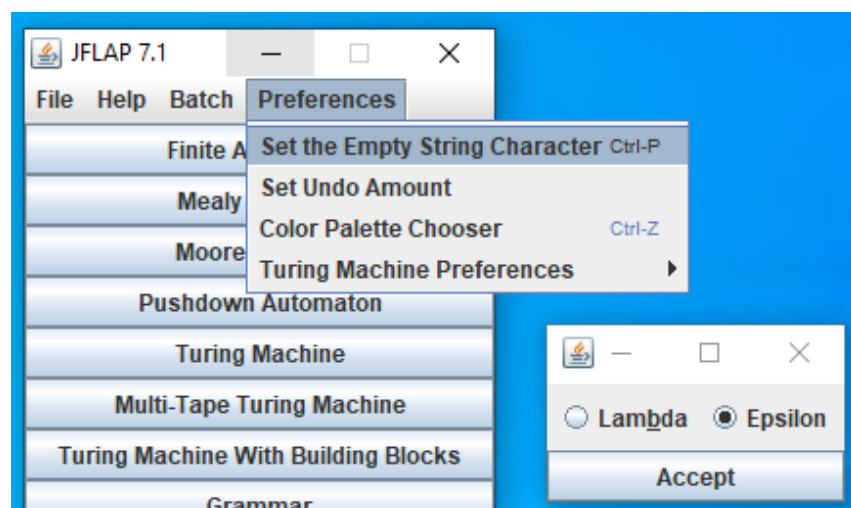
## JFLAP: graphical tools for Formal Languages and Automata Theory



Follow the JFLAP tutorial at <http://www.jflap.org/tutorial/fa/createfa/fa.html>. Then use JFLAP to draw and simulate some of the DFAs/NFAs discussed in the lecture.

Please note the following minor differences:

- An **accept** states is called a **final** state in JFLAP.
- By default, the empty string  $\varepsilon$  is called  $\lambda$  (lambda) in JFLAP. To change this use: Preferences > Set the Empty String Character > Epsilon



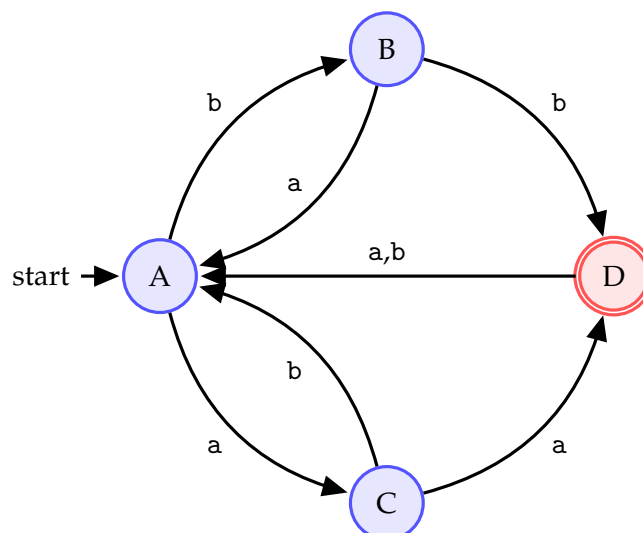
Optionally, you may want to read the first chapter from the “JFLAP Book” – available as PDF from the same folder. (Legal copy from <https://www2.cs.duke.edu/cs216/jflap/jflapbook/>).

*The aim of this exercise is to get you used to JFLAP and learn from experimenting with it. Play with it, develop your intuition and understanding of the concepts, experience the mistakes and errors, the failures and successes!*

*In particular, if something does not work then ask yourself: what is the source of the problem? How can I fix it?*

*If it works then ask: can I see the more general pattern? What are the transferable skills/knowledge I can use elsewhere?*

(2) Consider the following DFA:



Without using JFLAP, practice *simulating* the behaviour of this DFA using the following strings. (For each string, list the sequence of visited states, e.g.  $q_0, q_2, q_0, q_1, q_3, \dots$ ).

abba babb aaa bbabba

You can use the following table to organise your work:

Symbol	State	Symbol	State	Symbol	State	Symbol	State
	A		A		A		A
a		b		a		b	
b		a		a		b	
b		b		a		a	
a		b				b	
						b	
						a	

### Solution

One convenient way of listing the visited states is to use tables as follows:

a	A
b	C
b	A
b	B
a	A
	reject

b	A
a	B
a	A
b	B
b	D
	accept

a	A
a	C
a	D
a	A
	reject

b	A
b	B
b	D
a	A
b	B
b	D
a	A
	reject

Produce the formal definition of the above DFA. This should consist of: the alphabet  $\Sigma$ , the set of states  $Q$ , the transition function  $\delta$ , in table form, the start state, and the set of accept states  $F$ .

### Solution

$$\Sigma = \{a, b\}$$

$$Q = \{A, B, C, D\}$$

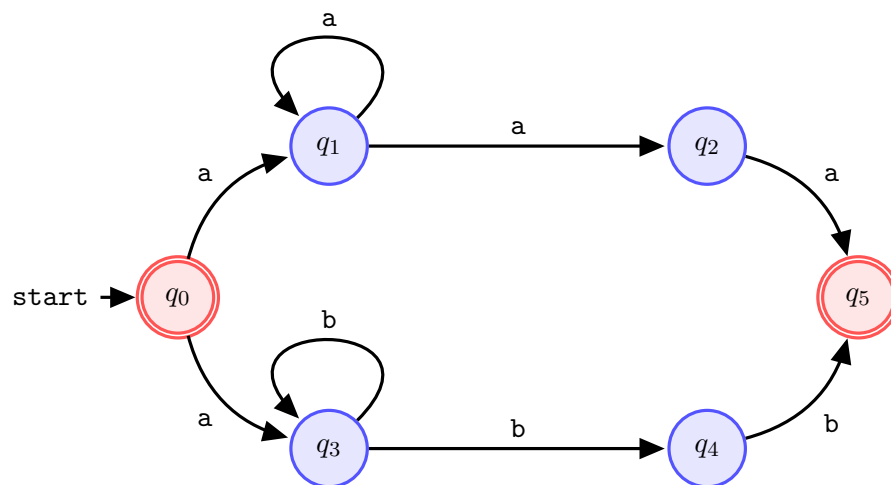
$$\delta :$$

	a	b
A	C	B
B	A	D
C	D	A
D	A	A

$$q_{\text{start}} = A$$

$$F = \{D\}$$

(3) Consider the following NFA:



Without using JFLAP, practice *simulating* the behaviour of this NFA using the following strings. (For each string, list the sets of visited states, e.g.  $\{q_0\}$ ,  $\{q_1, q_2\}$ ,  $\{q_2, q_3\}$ ,  $\dots$ ).

aa    aaaa    abbaa    babb    aaaba    abbbbbbbbaab

You may want to use a table similar to the one in the previous question.

### Solution

	$\{q_0\}$
a	$\{q_1, q_3\}$
a	$\{q_1, q_2\}$
	reject

	$\{q_0\}$
a	$\{q_1, q_3\}$
a	$\{q_1, q_2\}$
a	$\{q_1, q_2, q_5\}$
a	$\{q_1, q_2, q_5\}$
	accept

	$\{q_0\}$
a	$\{q_1, q_3\}$
b	$\{q_3, q_4\}$
b	$\{q_3, q_4, q_5\}$
a	$\emptyset$
a	$\emptyset$
	reject

	$\{q_0\}$
a	$\{q_1, q_3\}$
b	$\{q_3, q_4\}$
b	$\{q_3, q_4, q_5\}$
b	$\{q_3, q_4, q_5\}$
b	$\{q_3, q_4, q_5\}$
b	$\{q_3, q_4, q_5\}$
b	$\{q_3, q_4, q_5\}$
b	$\{q_3, q_4, q_5\}$
a	$\emptyset$
a	$\emptyset$
b	$\emptyset$
	reject

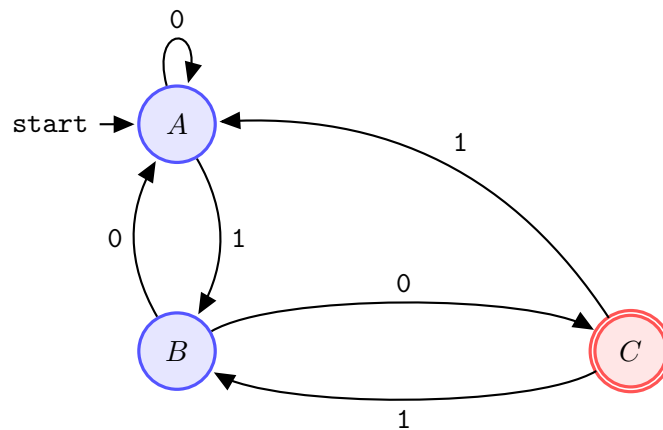
## Solution

$\delta :$

	a	b
$\rightarrow *q_0$	$\{q_1, q_3\}$	$\emptyset$
$q_1$	$\{q_1, q_2\}$	$\emptyset$
$q_2$	$\{q_5\}$	$\emptyset$
$q_3$	$\emptyset$	$\{q_3, q_4\}$
$q_4$	$\emptyset$	$\{q_5\}$
$*q_5$	$\emptyset$	$\emptyset$

$$F = \{q_5\}$$

- (4) Which of the strings listed below does the following NFA accept? (Use pen-and-paper, not JFLAP)



- 10011010
- 01010011
- 00010111
- 0010010

### Solution

Only the last string, 0010010, is accepted.

- (5) The formal description  $(Q, \Sigma, \delta, q_{\text{start}}, F)$  of a DFA is given by

$$(\{q_1, q_2, q_3, q_4, q_5\}, \{d, u\}, \delta, q_1, \{q_5\}),$$

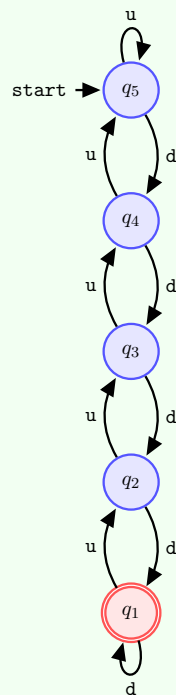
where  $\delta$  is given by the following table

	d	u
* $q_1$	$q_1$	$q_2$
$q_2$	$q_1$	$q_3$
$q_3$	$q_2$	$q_4$
$q_4$	$q_3$	$q_5$
$\rightarrow q_5$	$q_4$	$q_5$

Give the *state diagram* of this machine.

**Hint:** It looks like a ladder or a lift going between floors. (u: go up, d: go down.)

## Solution

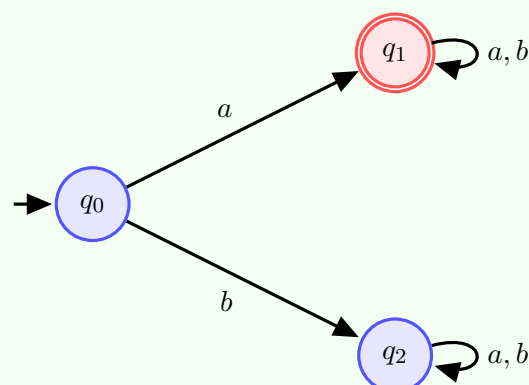


(6) Use JFLAP to design simple DFAs which recognize the following languages over the alphabet  $\Sigma = \{a, b\}$

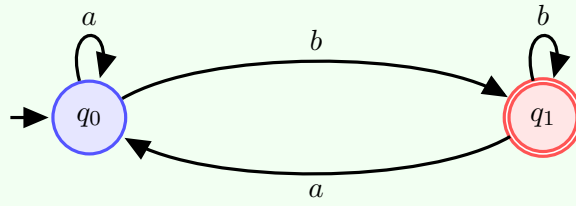
- 1) The language of strings which begin with  $a$ .
- 2) The language of strings which end with  $b$ .
- 3) The language of strings which either begin **or** end with  $b$ .
- 4) The language of strings which begin with  $a$  **and** end with  $b$ .
- 5) The language of strings which contain the substring  $ba$ .
- 6) The language of strings with all the  $a$ 's on the left and  $b$ 's on the right
- 7) The language strings consisting of alternating  $a$ 's and  $b$ 's.

## Solution

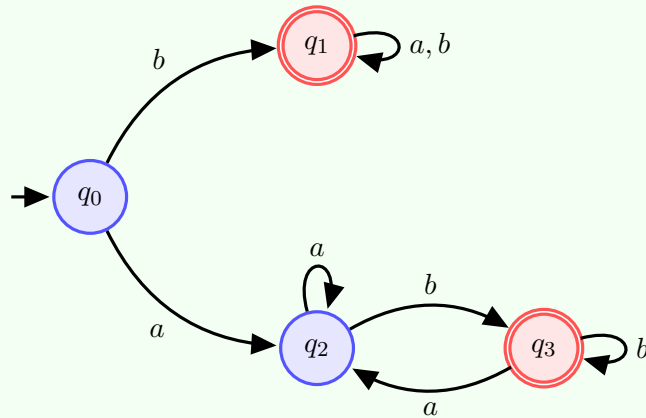
1)



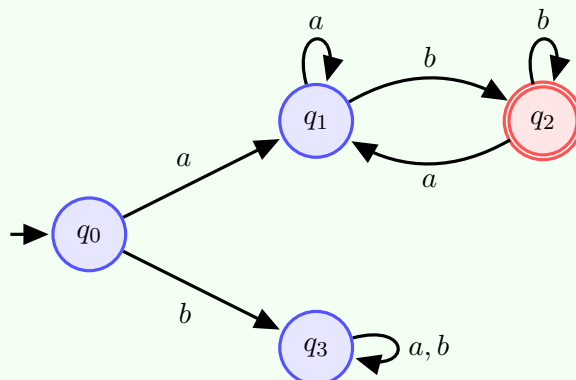
2)



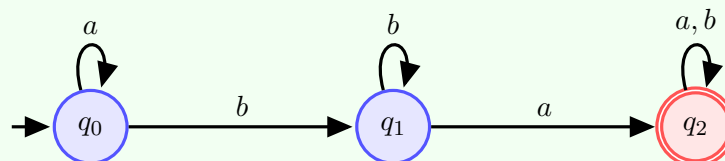
3)



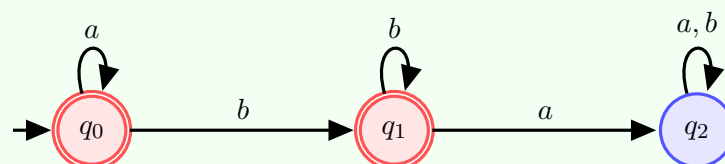
4)



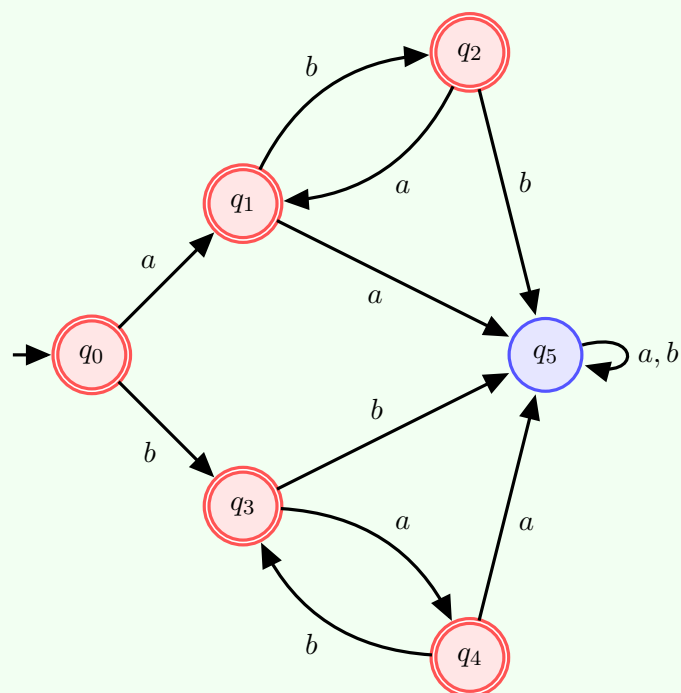
5)



6)

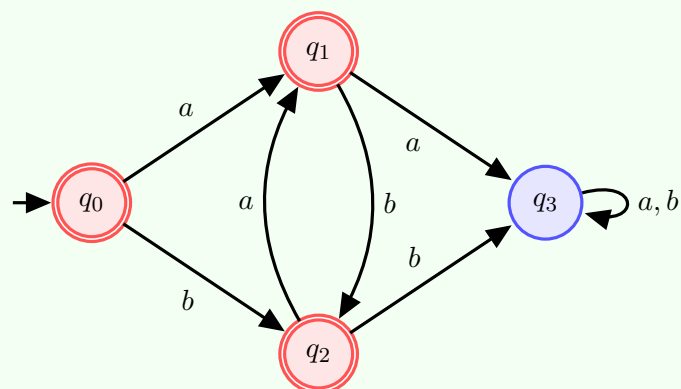


7) Assume that  $\varepsilon$  (the empty string) also satisfies the required property.



$q_5$  here only plays the role of a “trap” state.

A simpler DFA is:





(7) Use JFLAP to produce NFAs to recognize the following languages over  $\Sigma = \{0, 1\}$

- 1) The language of strings which begin and end with 01.
- 2) The language of strings which do **not** end with 01.

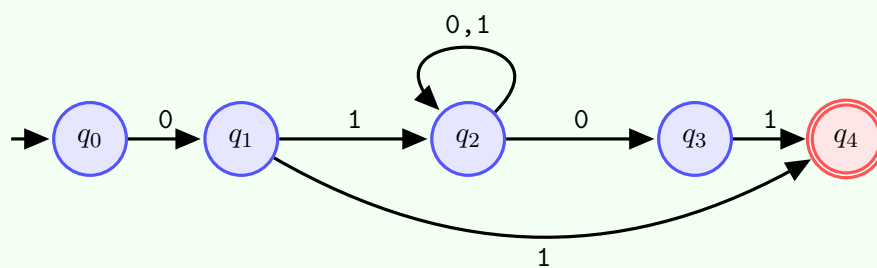
**Hint:** Recall that DFAs are a special case of NFAs. For this problem, it may be easier to first create a DFA that accepts *strings that end with 01*, then we flip the accepting states into non-accepting ones, and vice versa. This will produce a DFA that accepts *strings that do not end with 01*, as required.

This is a useful technique, but note that **it only works for DFAs – it does not work for NFAs**.

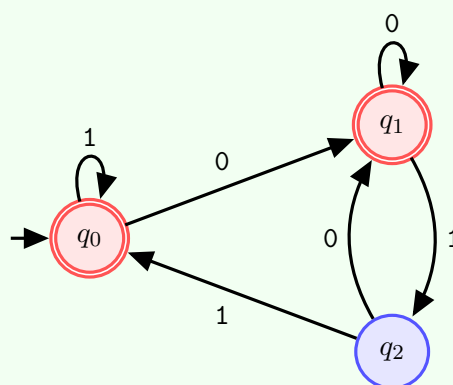
- 3) The language of strings which begin and end with different symbols.
- 4) The language of strings of odd length.
- 5) The language of strings which contain an even number of 0's.
- 6) The language of binary numbers which are divisible by 4.

### Solution

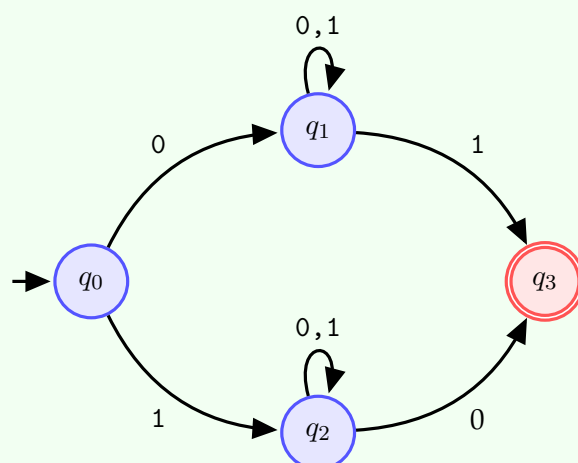
- 1) Note that the statement also allows for the string 01 to be accepted.



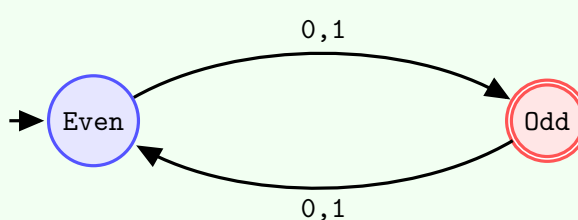
- 2)



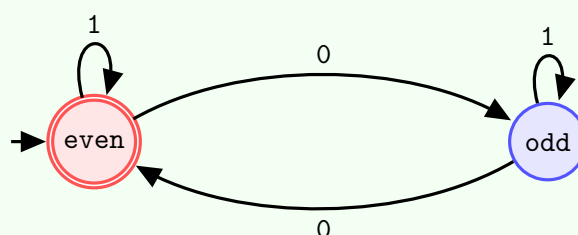
3)



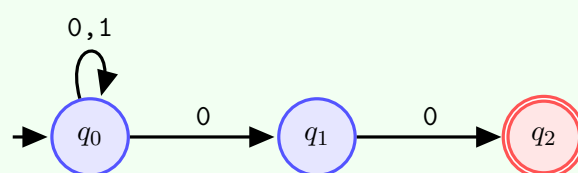
4)



5)



6) A number is divisible by 4 if its binary representation ends with 00.



- (8) If  $a$  is a *symbol* from an alphabet  $\Sigma$  then  $a^n$  denotes the string which consists of  $n$  successive copies of  $a$ .

Similarly, if  $x$  is a *string* of symbols then  $x^n$  denotes the string which consists of  $n$  successive copies of  $x$ . For example,  $a^2 = aa$  and  $(ab)^2 = abab$ .

Let  $\Sigma = \{0, 1\}$ . Write  $0^4$ ,  $1^4$ ,  $(10)^3$ ,  $10^3$  explicitly as strings in the usual form.

### Solution

$$\begin{aligned}
 0^4 &= 0000 \\
 1^4 &= 1111 \\
 (10)^3 &= 101010 \\
 10^3 &= 1000
 \end{aligned}$$

- (9) If  $\Sigma$  is an alphabet then  $\Sigma^n$  denotes the set of all strings over  $\Sigma$  which have length

exactly  $n$  symbols.

1) Let  $\Sigma = \{a, b, c\}$ . Find  $\Sigma^2$ .

2) Let  $\Sigma = \{a, b\}$ . Find  $\Sigma^3$ .

### Solution

1) For  $\Sigma = \{a, b, c\}$ :

$$\Sigma^2 = \{aa, ab, ac, ba, bb, bc, ca, cb, cc\}$$

2) For  $\Sigma = \{a, b\}$ :

$$\Sigma^3 = \{aaa, aab, aba, baa, bba, bab, abb, bbb\}$$

(10) If  $\Sigma$  is an alphabet then the set of all finite-length strings over it is denoted by  $\Sigma^*$ .

Let  $\Sigma_1 = \{a\}$  and  $\Sigma_2 = \{a, b\}$ . List the strings of length 0, 1, 2, 3, and 4 over these two alphabets. Write these in the form  $\Sigma_1^* = \{\dots\}$  and  $\Sigma_2^* = \{\dots\}$ .

Note that

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \Sigma^4 \cup \dots$$

### Solution

For  $\Sigma_1 = \{a\}$ :

Length	Strings
0	$\varepsilon$
1	a
2	aa
3	aaa
4	aaaa

So,

$$\Sigma_1^* = \{\varepsilon, a, aa, aaa, aaaa, \dots\}$$

For  $\Sigma_2 = \{a, b\}$ :

Length	Strings
0	$\varepsilon$
1	a b
2	aa ab ba bb
3	aaa aab aba abb baa bab bba bbb
4	aaaa aaab aaba aabb abaa abab abba abbb baaa baab baba babb bbba bbab bbbb

So,

$$\begin{aligned} \Sigma_2^* = \{ & \varepsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, bab, bba, bbb, \\ & aaaa, aaab, aaba, aabb, abaa, abab, abba, abbb, \\ & baaa, baab, baba, babb, bbba, bbab, bbbb, \dots \} \end{aligned}$$

**(1) Intersection, union and difference of DFAs**

Given two languages described by two DFAs, the idea is to construct a new DFA that simulates simultaneously-running the given DFAs. To do this, the states of the new DFA will be pairs of states from the original DFAs.

Suppose  $\mathcal{M}_1 = (Q_1, \Sigma, \delta_1, q_{\text{start } 1}, F_1)$  and  $\mathcal{M}_2 = (Q_2, \Sigma, \delta_2, q_{\text{start } 2}, F_2)$  are DFAs recognizing  $L_1$  and  $L_2$ , respectively.

Let  $\mathcal{M}$  be the DFA given by  $(Q, \Sigma, \delta, q_0, F)$  where

$$\begin{aligned} Q &= Q_1 \times Q_2 \\ q_0 &= (q_{\text{start } 1}, q_{\text{start } 2}) \end{aligned}$$

and the transition function  $\delta$  is defined by

$$\delta((p, q), \sigma) = (\delta_1(p, \sigma), \delta_2(q, \sigma))$$

for all  $(p, q) \in Q_1 \times Q_2$  and  $\sigma \in \Sigma$ .

Then

- $\mathcal{M}$  recognizes  $L_1 \cap L_2$  if

$$F = \{(p, q) \mid p \in F_1 \wedge q \in F_2\} = F_1 \times F_2$$

- $\mathcal{M}$  recognizes  $L_1 \cup L_2$  if

$$\begin{aligned} F &= \{(p, q) \mid p \in F_1 \vee q \in F_2\} \\ &= \{(p, q) \mid p \in F_1\} \cup \{(p, q) \mid q \in F_2\} \\ &= (F_1 \times Q_2) \cup (Q_1 \times F_2) \end{aligned}$$

- $\mathcal{M}$  recognizes  $L_1 - L_2$  if

$$F = \{(p, q) \mid p \in F_1 \wedge q \notin F_2\} = F_1 \times (Q_2 - F_2)$$

1) Let the languages  $L_1$  and  $L_2$  be given by:

- $L_1 = \{w \mid w = a^n b v \text{ where } n \geq 0 \text{ and } v \in \Sigma^*\}$
- $L_2 = \{w \mid w = a^n b v \text{ where } n \geq 0 \text{ and } v \in \Sigma^*\}$

over  $\Sigma = \{a, b\}$ . Recall that  $\Sigma^*$  is the set of all strings over  $\Sigma$  of finite length.

First, design DFAs recognising  $L_1$  and  $L_2$ , then use the above method to construct a DFA for

$$L_1 \cap L_2 = \{w \mid w \text{ has at least one } a \text{ and at least one } b\},$$

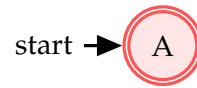
then outline how it can easily be changed for  $L_1 \cup L_2$  and  $L_1 - L_2$ .

2) Use the same method for the language

$$\{w \mid w \text{ has an even number of } a\text{'s and each } a \text{ is followed by at least one } b\}.$$

**(2) (Complement of a language)**

In the special case where  $L_1 = \Sigma^*$ , the language of all possible strings of finite length over the given alphabet  $\Sigma$ , we have the following NFA



i.e.  $\mathcal{M}_1 = (Q_1, \Sigma, \delta_1, q_{\text{start } 1}, F_1)$  where

$$\begin{aligned} Q_1 &= \{A\} \\ \Sigma &= \Sigma \\ \delta_1 &: (q, \sigma) \mapsto \{A\} \\ q_{\text{start } 1} &= A \\ F_1 &= \{A\} \end{aligned}$$

This choice for  $\mathcal{M}_1$  provides us with a method to produce the **complement** of  $L_2$  which is defined as  $\Sigma^* - L_2 = L_1 - L_2$ . Following the result from the previous question, let

$$\begin{aligned} Q &= Q_1 \times Q_2 = \{q_{\text{start } 1}\} \times Q_2 \\ F &= F_1 \times (Q_2 - F_2) = \{q_{\text{start } 1}\} \times (Q_2 - F_2) \end{aligned}$$

This means that the new DFA is essentially the **DFA for  $L_2$  but with the accepting and non-accepting states “flipped.” (swapping roles.)**

Use this observation to produce a DFA for the language

$$\{w \mid w \text{ does **not** contain the substring baba.}\}$$

*This does not always work for NFAs – give an example to show that it does not work. (Counter example)*

**Solution**

Sketch: First create a DFA that accepts the strings that do **not** satisfy the required property, then flip the accepting states into non-accepting ones, and vice versa.

This will produce a DFA that accepts the required strings.

In your preferred programming language, implement a class for representing and simulating DFAs and NFAs.