Basic

(1) 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.

**Solution**

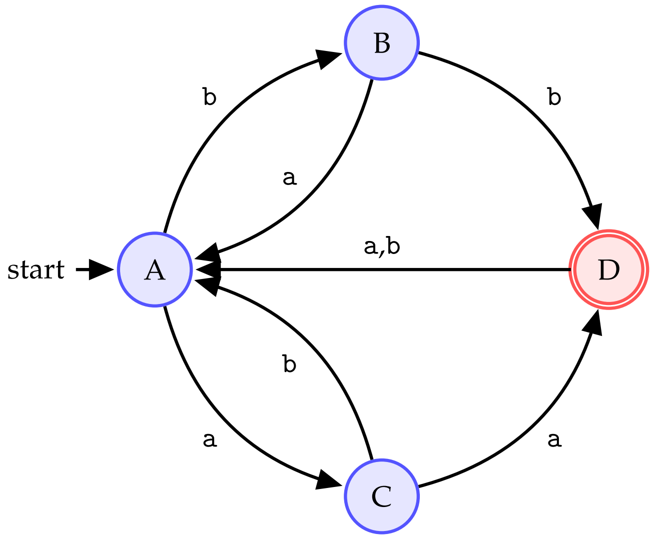
*The aim of this exercise is to get used to JFLAP and learn from experimenting with it.*

*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 what are the transferable skills/knowledge I can use elsewhere?*

(2) Consider the following DFA:




Without using JFLAP, practice *simulating* the behaviour of ths DFA using the following strings

`abba` `babb` `aaa` `bbabba`

For each string, list the sequence of states visited by this DFA (e.g. $q_0, q_2, q_0, q_1, q_3, \ldots$).

**Solution**

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

| | |
|---|---|
| | A |
| a | C |
| b | A |
| b | B |
| a | A |
| | reject |

| | |
|---|---|
| | A |
| b | B |
| a | A |
| b | B |
| b | D |
| | accept |

| | |
|---|---|
| | A |
| a | C |
| a | D |
| a | A |
| | reject |

| | |
|---|---|
| | 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 final states $F$.

**Solution**

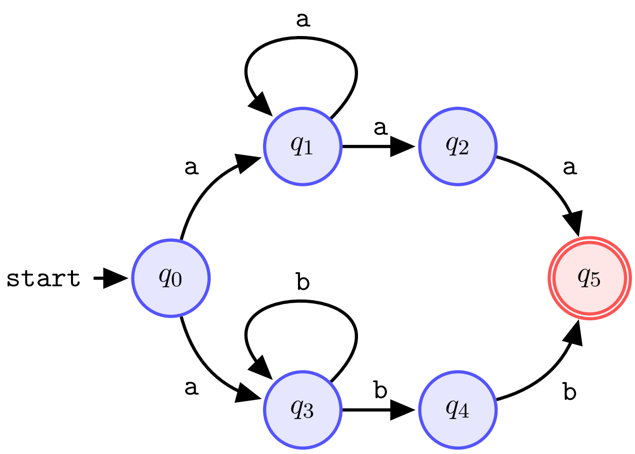
$$\Sigma = \{\mathtt{a}, \mathtt{b}\}$$
$$Q = \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}\}$$

$\delta$ :

| $Q$ | a | b |
|---|---|---|
| A | C | B |
| B | A | D |
| C | D | A |
| D | A | A |

$$q_{\text{start}} = \mathrm{A}$$
$$F = \{\mathrm{D}\}$$

(3) Consider the following NFA:




Without using JFLAP, practice *simulating* the behaviour of this NFA using the following strings.

`abbaa` `babb` `aaaba` `abbbbbbbaab`

For each string, list the sets of states visited by the NFA (e.g. $\{q_0\}, \{q_1, q_2\}, \{q_2, q_3\}, \ldots$).

**Solution**

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

| | |
|---|---|
| | $\{q_0\}$ |
| b | $\emptyset$ |
| a | $\emptyset$ |
| b | $\emptyset$ |
| b | $\emptyset$ |
| | reject |

| | |
|---|---|
| | $\{q_0\}$ |
| a | $\{q_1, q_3\}$ |
| a | $\{q_1, q_2\}$ |
| a | $\{q_1, q_2, q_5\}$ |
| b | $\emptyset$ |
| 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 |

Produce the formal definition $(\Sigma, Q, \delta, q_{\text{start}}, F)$ of the above NFA.

**Solution**

$$\Sigma = \{\mathtt{a}, \mathtt{b}\}$$
$$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$$

$\delta$ :

| $Q$ | a | b |
|---|---|---|
| $q_0$ | $\{q_1, q_3\}$ | $\emptyset$ |
| $q_1$ | $\emptyset$ | $\{q_1, q_2\}$ |
| $q_2$ | $\{q_5\}$ | $\emptyset$ |
| $q_3$ | $\emptyset$ | $\{q_3, q_4\}$ |
| $q_4$ | $\emptyset$ | $\{q_5\}$ |
| $q_5$ | $\emptyset$ | $\emptyset$ |

$$q_{\text{start}} = q_0$$
$$F = \{q_5\}$$

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

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

where $\delta$ is given by the following table

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

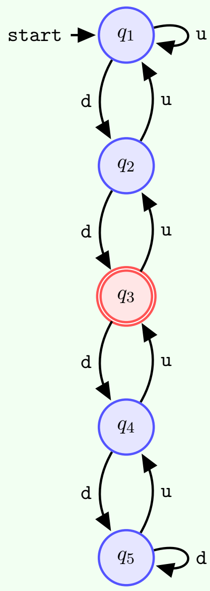
Give the *state diagram* of this machine.

**Solution**

**Hint:**

It looks like a ladder or a lift going between floors.

(u: go up, d: go down.)



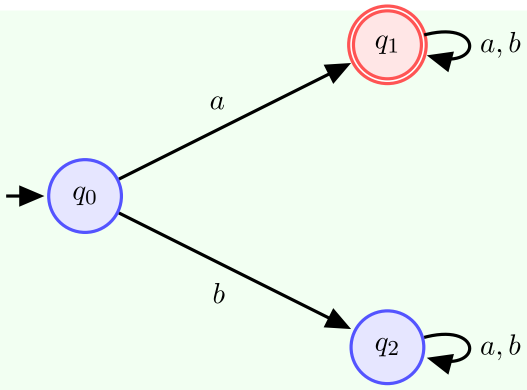

(5) Use JFLAP to design simple DFAs which recognize the following languages over the alphabet $\Sigma = \{a, b\}$

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.

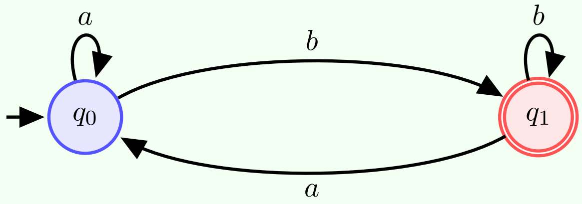
**Solution**

a)

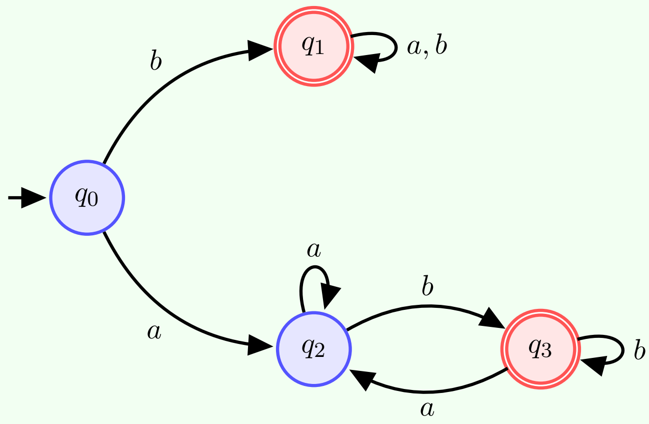
Basic



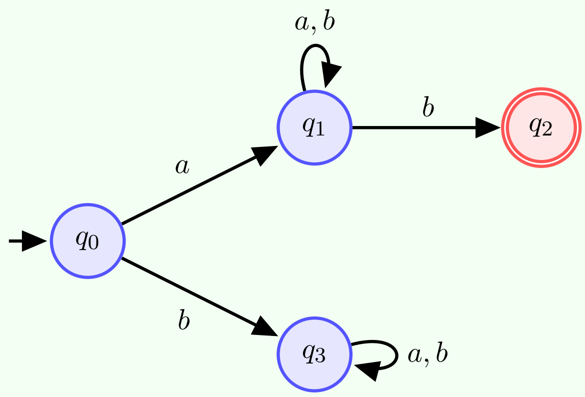

b)



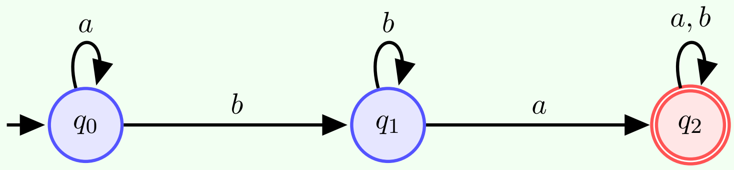

c)



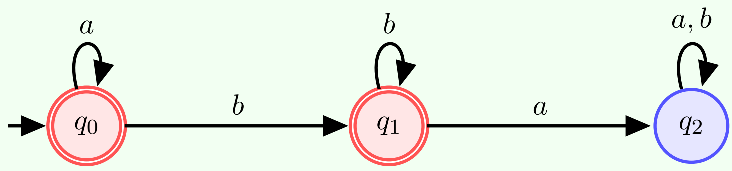
d)



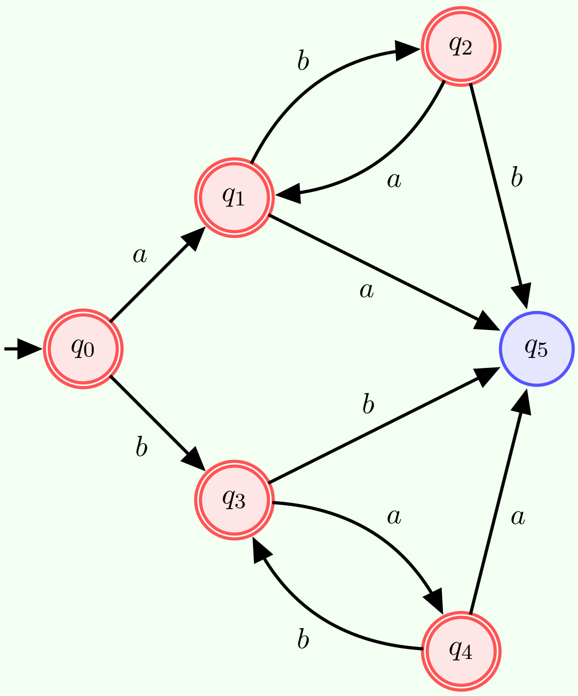

e)




f)

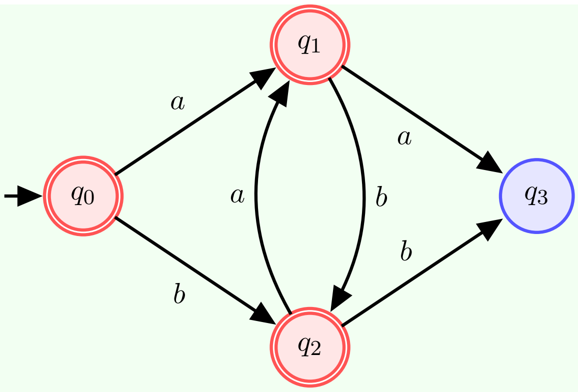



g) Assume that $\varepsilon$ (the empty string) also satisfies the required property.




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

A simpler DFA is:

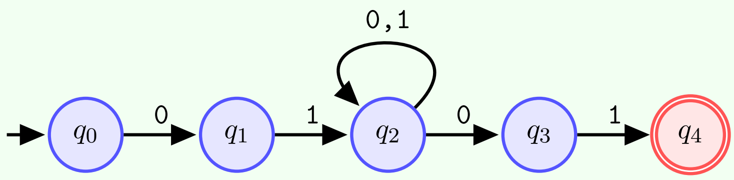



(6) Use JFLAP to produce NFAs to recognize the following languages over $\Sigma = \{0, 1\}$

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

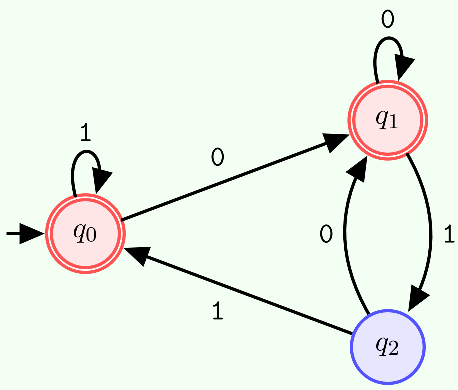
**Solution**

a)

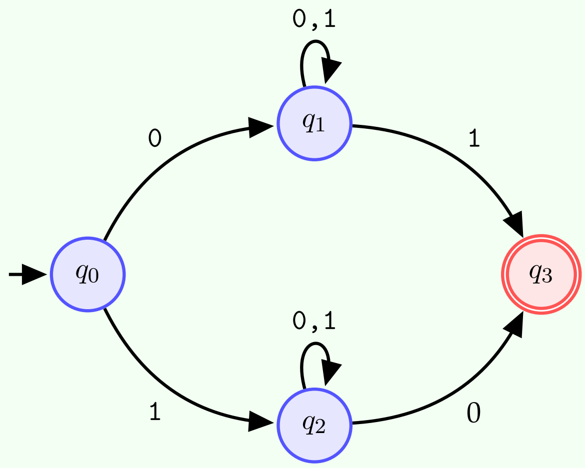



b) Recall that DFAs are a special case of NFAs. For this problem, it may be easier to first create a DFA that accepts *strings that **end** with* `01`, then we flip the accepting states into non-accepting ones, and vice versa. This will produce a DFA that accepts *strings that **do not end** with* `01`,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

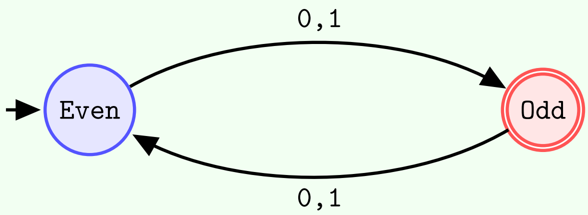
This is a useful technique, but note that **it only works for DFAs – it does not work for NFAs.**



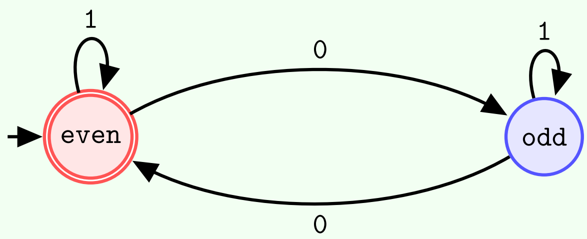

c)



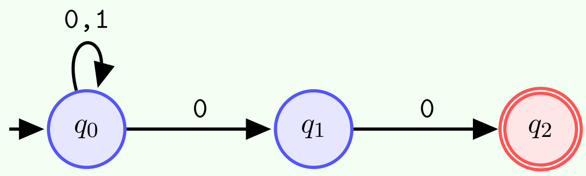

d)




e)




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



Basic

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

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

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

**Solution**

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

(8) If $\Sigma$ is an alphabet then $\Sigma^n$ denotes the set of all strings over $\Sigma$ which have length exactly $n$ symbols.

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

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

**Solution**

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

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

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

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

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

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

Note that

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

**Solution**

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

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

So,

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

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

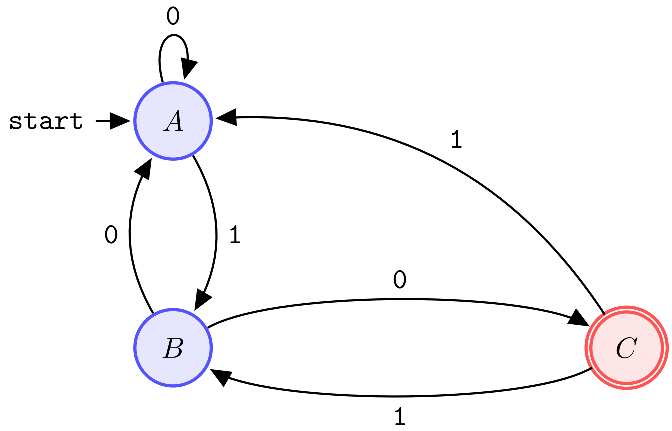
| Length | Strings | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | $\varepsilon$ | | | | | | | |
| 1 | a | b | | | | | | |
| 2 | aa | ab | ba | bb | | | | |
| 3 | aaa | aab | aba | abb | baa | bab | bba | bbb |
| 4 | aaaa | aaab | aaba | aabb | abaa | abab | abba | abbb |
| | baaa | baab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

So,

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

Intermediate

(1) Which of the strings listed below does the following NFA accept?




- `10011010`
- `01010011`
- `00010111`
- `0010010`

(2) **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

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

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 $F = \{(p,q) \mid p \in F_1 \vee q \in F_2\} = (F_1 \times Q_2) \cup (Q_1 \times F_2)$
- $\mathcal{M}$ 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 the two RegEx's: $\mathtt{b}^*\mathtt{a}\Sigma^*$ and $\mathtt{a}^*\mathtt{b}\Sigma^*$, respectively, where $\Sigma = \{\mathtt{a}, \mathtt{b}\}$.

   First, produce state diagram for DFAs recognizing $L_1$ and $L_2$, then use the above method to construct a DFA for

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

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

2) Use the same method for the language

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

Intermediate

(3) **(Complement of a language)**

In the special case where $L_1 = \mathcal{L}(\Sigma^*)$, i.e. the language of all possible strings over $\Sigma$, we get

$$\mathcal{M}_1 = (Q_1, \Sigma, \delta_1, q_{0_1}, F_1) = \Big(\{q_{0_1}\}, \Sigma, (q, \sigma) \mapsto q_{0_1}, q_{0_1}, \{q_{0_1}\}\Big)$$

This choice for $\mathcal{M}_1$ provides us with a method to produce the **complement** of $L_2$ which is defined as $\mathcal{L}(\Sigma^*) - L_2 = L_1 - L_2$.

Now, note that

$$\begin{aligned} Q &= Q_1 \times Q_2 = \{q_{0_1}\} \times Q_2 \\ F &= F_1 \times (Q_2 - F_2) = \{q_{0_1}\} \times (Q_2 - F_2) \end{aligned}$$

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

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

*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.