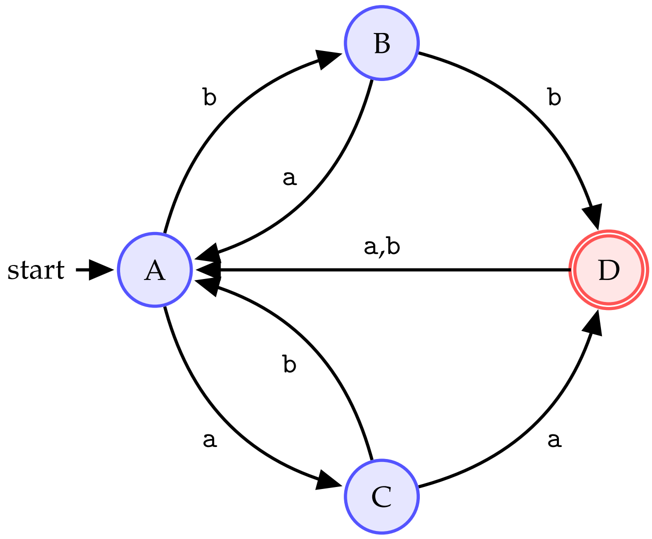
(1) Follow the JFLAP tutorial at http://www.jflap.org/tutorial/fa/createfa/fa.html. Then use JFLAP to draw and simulate some of the DFAs/NFAs discussed in the lecture.

(2) Consider the following DFA:



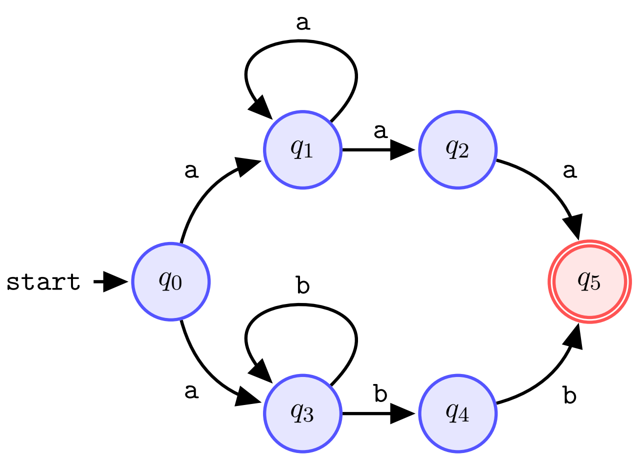
Without using JFLAP, practice *simulating* the behaviour of ths DFA using the following strings

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

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

Produce the formal definition of the above DFA. This should consist of: the alphabet Σ, the set of states Q, the transition function δ, in table form, the start state, and the set of final states F.

(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. $\{q_0\}, \{q_1, q_2\}, \{q_2, q_3\}, \ldots$).

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

Basic

(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 δ is given by the following table

	u	d
$\to q_1$	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.

(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.

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

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.

(7) If $\mathtt{a}$ is a *symbol* from an alphabet Σ 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.

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

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

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

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

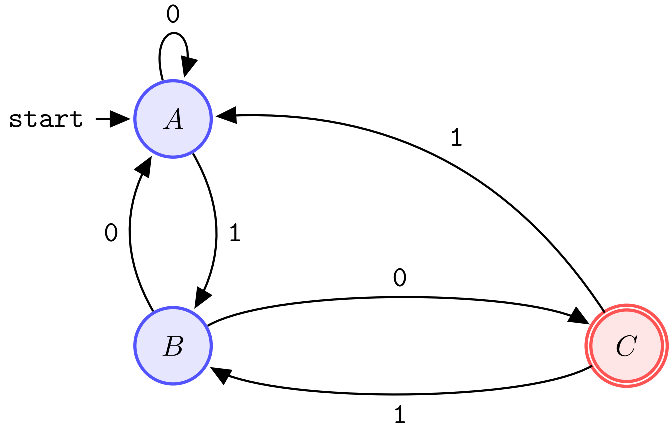
Let $\Sigma_1 = \{\mathtt{a}\}$ and $\Sigma_2 = \{\mathtt{a}, \mathtt{b}\}$. List the strings of length 0, 1, 2, 3, and 4 over these two alphabets. Write these in the form $\Sigma_1^* = \{\ldots\}$ and $\Sigma_2^* = \{\ldots\}$.

Note that

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

Basic

(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 δ 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 Σ, 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)