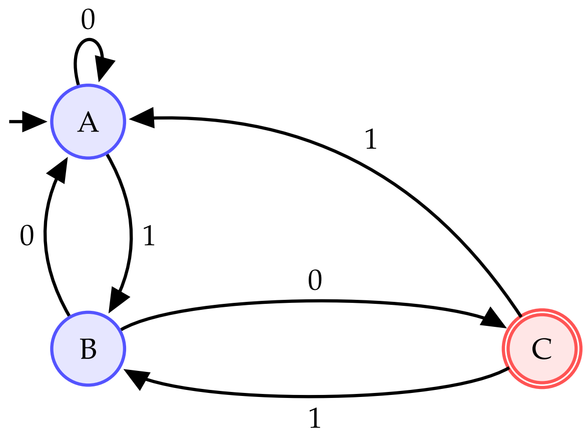
(1) Consider the following NFA.



1) What is its transition table?

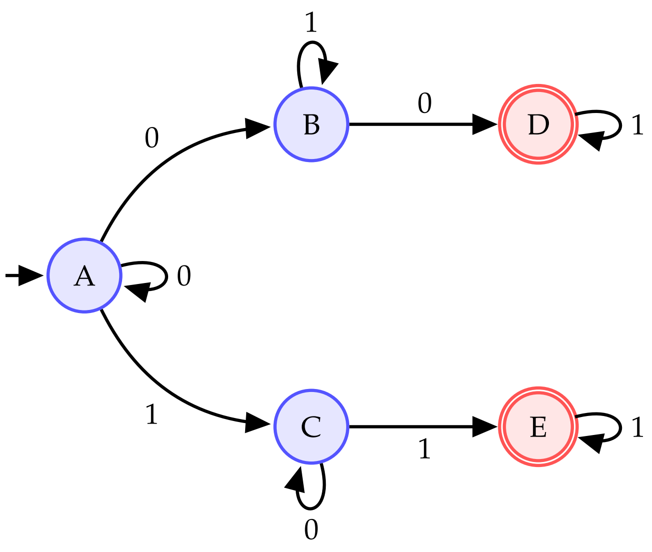
2) Use the *subset construction method* to convert it to an equivalent DFA.

3) Draw the state diagram of the resulting DFA.

4) Which of the following sets of NFA states is not a state of the DFA that is reachable from the start state of the DFA?

- $\{A, C\}$
- $\{B, C\}$
- $\{A\}$
- $\{B\}$

(2) Use the *subset construction method* to convert the following NFA to an equivalent DFA.



		0	1
→	A	$\{A, B\}$	$\{C\}$
	B	$\{D\}$	$\{B\}$
	C	$\{C\}$	$\{E\}$
*	D	$\emptyset$	$\{D\}$
*	E	$\emptyset$	$\{E\}$

Hint: The resulting DFA has 13 states, 8 of which are accepting states.

(3) Design an NFA that accepts strings over $\{\texttt{a}, \texttt{b}\}$ which end with `aaa`, then convert it to an equivalent DFA.

(4) Design an NFA that accepts strings over $\{\texttt{0}, \texttt{1}\}$ which have `1` in the second position from the end (e.g. `0010`, `1011`,`10`, etc.), then convert it to an equivalent DFA.

(1) **(Optional exercise – Regular Expressions in practice)**

Suppose we have a programming language where comments are delimited by `@=` and `=@`. Let L be the language of all valid delimited comment strings, i.e. a member of L must begin with `@=` and end with `=@`.

Use the page at `https://regex101.com/r/Ez1kqp/3` and try the following RegEx searches:

Programming	380CT notation	Interpreation
`@`	@	Just the symbol @
`@=`	@=	Just the string @=
`.`	Σ	Any symbol from the alphabet
`.*`	Σ^*	Any string over the alphabet
`@.*`	@Σ^*	
`@.*|.*@`	@Σ^* + Σ^*@	
`@.*@`	@Σ^*@	
`@=.*=@`	@=Σ^*=@	

Interpret the results for the last 4 searches. Try alternative searches to develop your understanding of how RegEx is used in practice. What is the correct RegEx for L?

N.B. Please note that the regular expressions used in programming languages are more general than RegEx's defined for Regular Languages.
See for example `https://en.wikipedia.org/wiki/RE2_(software)`

(2) Complete the descriptions of the following regular expressions (write in the shaded boxes). Assume the alphabet $\Sigma = \{0, 1\}$ in all the parts.

Recall that, unless brackets are used to explicitly denote precedence, the **operators precendence** for the regular operations is: *star, concatenation*, then *union*.

1) $01 + 10 = \{\boxed{}, \boxed{}\}$
2) $(\varepsilon + 0)(\varepsilon + 1) = \{\varepsilon, 0, 1, \boxed{}\}$
3) $(0 + \varepsilon)1^* = 01^* + 1^* = \{w \mid w$ has at most $\boxed{}$ and is at the start of $w\}$
4) $\Sigma^*0 = \{w \mid w$ ends with a $\boxed{}\} = \{w \mid w$ respresents an $\boxed{}$ number in binary$\}$
5) $0^*10^* = \{w \mid w$ contains a single $\boxed{}\}$
6) $\Sigma^*0\Sigma^* = \{w \mid w$ has at least one $\boxed{}\}$
7) $\Sigma^*001\Sigma^* = \{w \mid w$ contains the string $\boxed{}$ as a substring$\}$
8) $\Sigma^*000^*\Sigma^* = \{w \mid w$ cotains at least $\boxed{}$ consective $\boxed{}$'s$\}$
9) $(011^*)^* = \{w \mid$ every $\boxed{}$ in w is followed by at least one $\boxed{}\}$
10) $\Sigma\Sigma + \Sigma\Sigma\Sigma = \Sigma\Sigma(\varepsilon + \Sigma) = \{w \mid$ the length of w is exactly $\boxed{}$ or $\boxed{}\}$
11) $(\Sigma\Sigma)^* = \{w \mid w$ is a string of $\boxed{}$ length$\}$
12) $(\Sigma\Sigma\Sigma)^* = \{w \mid$ the length of w is a multiple of $\boxed{}\}$
13) $0\Sigma^*0 + 1\Sigma^*1 + 0 + 1 = \{w \mid w$ starts and ends with the $\boxed{}$ symbol$\}$

(3) Produce a *regular expression* for the following languages over the alphabet $\{\mathtt{a}, \mathtt{b}\}$

1) The language $L_\mathtt{a}$ of all strings that start with a.
2) The language $L_\mathtt{b}$ of all strings that end with b.
3) The union $L_\mathtt{a} \cup L_\mathtt{b}$.
4) The concatenation $L_\mathtt{a} L_\mathtt{b}$.
5) $L = (L_\mathtt{a} \cup L_\mathtt{b}) L_\mathtt{a} L_\mathtt{b}$.
6) The star closure of L: L^*.

Produce ε-NFAs for each of the above using the constructions shown in the lecture for the union, concatenation, and star.

(4) For each of the following RegEx's, give two strings that are members of the corresponding language, and two strings that are not. (A total of 4 strings for each part.)

Assume the alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ in all the parts.

1) $\mathtt{a}^*\mathtt{b}^*$
2) $\mathtt{a}(\mathtt{ba})^*\mathtt{b}$
3) $\mathtt{a}^* + \mathtt{b}^*$
4) $(\mathtt{aaa})^*$
5) $\Sigma^*\mathtt{a}\Sigma^*\mathtt{b}\Sigma^*\mathtt{a}\Sigma^*$
6) $\mathtt{aba} + \mathtt{bab}$
7) $(\varepsilon + \mathtt{a})\mathtt{b}$
8) $(\mathtt{a} + \mathtt{ba} + \mathtt{bb})\Sigma^*$

(5) Give regular expressions generating the languages below over $\Sigma = \{0, 1\}$

1) $\{w \mid w$ begins with 1 and ends with a 0$\}$
2) $\{w \mid w$ contains at least three 1's$\}$
3) $\{w \mid w$ contains the substring 0101$\}$
4) $\{w \mid w$ has length at least 3 and its third symbol is 0$\}$
5) $\{w \mid w$ starts with 0 and has odd length, **or** starts with 1 and has even length$\}$
6) $\{w \mid w$ does not contain the substring 110$\}$
7) $\{w \mid$ the length of w is at most 5$\}$
8) $\{w \mid w$ is any string except 11 and 111$\}$
9) $\{w \mid$ every odd position of w is 1$\}$
10) $\{w \mid w$ contains at least two 0's and at most one 1$\}$
11) $\{\varepsilon, 0\}$
12) $\{w \mid w$ contains an even number of 0's, or contains exactly two 1's$\}$
13) The empty set.
14) All strings except the empty string.

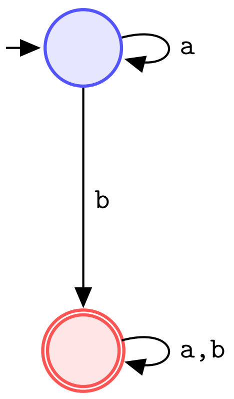
Reminder: We can convert any NFA into a GNFA as follows:

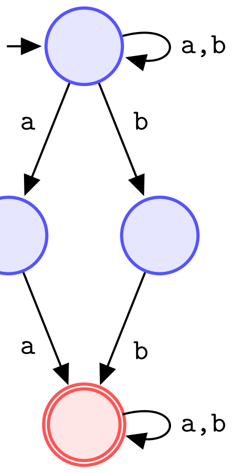
- Add a new start state with an ε-transition to the NFA's start state.
- Add a new accept state with ε-transitions from the NFA's accept states.
- If a transition has multiple labels then replace them with their union. (e.g. $a, b \to a + b$.)

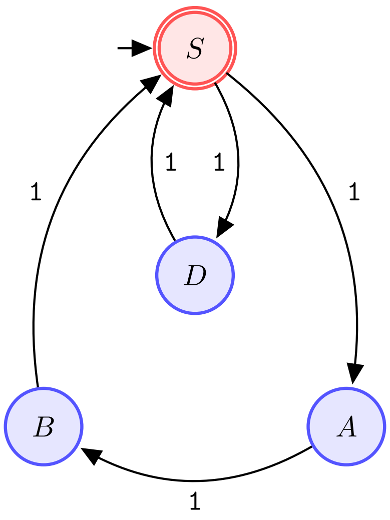
Once the GNFA is produced, start removing states, one at a time, and "patch" any affected transitions using regular expressions (RegEx's). Repeat until only two states (initial and accept) remain. The RegEx on the only remaining transition is the equivalent RegEx to the NFA.

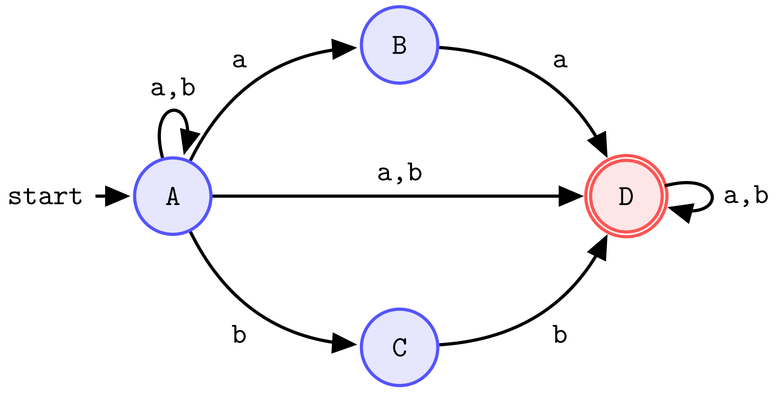
(1) Use the GNFA algorithm to find regular expressions for the languages recognized by the following NFAs.

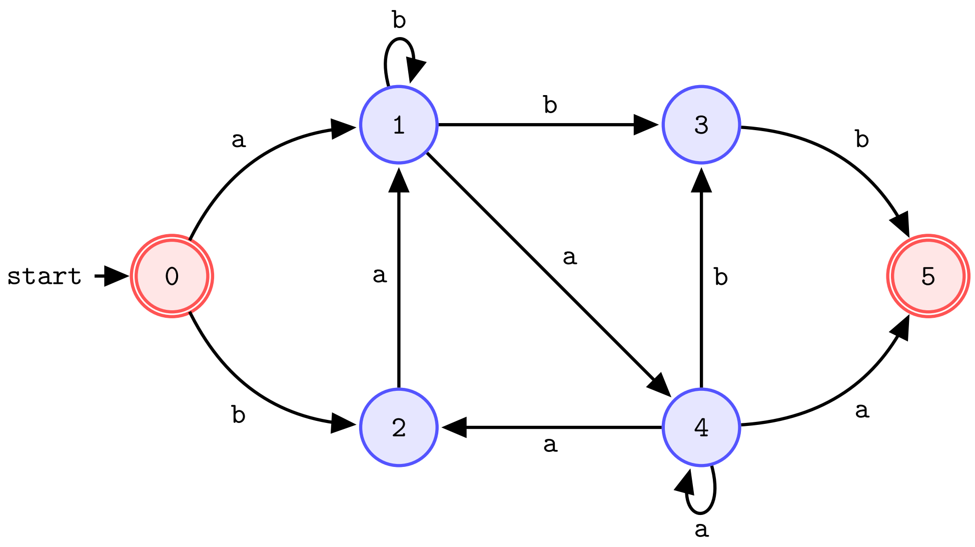
Can you interpret the results?



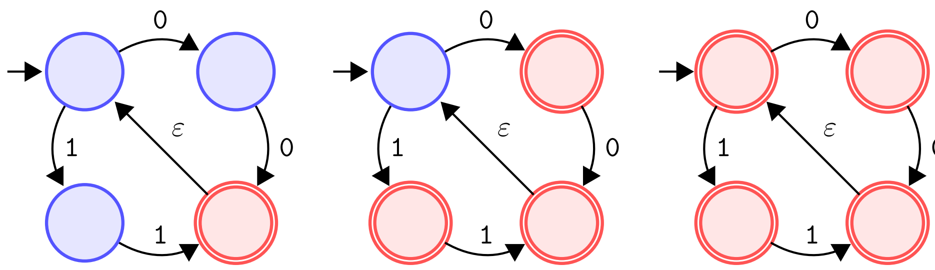








(2) Give RegEx's for the languages recognized by the following similar NFAs, using the GNFA algorithm. What do you notice?



(3) Let L_n be the language of all strings over $\Sigma = \{1\}$ that have length a multiple of n, where n is a natural number (i.e. $n \in \mathbb{N} = \{1, 2, 3, \ldots\}$).

1) Design an NFA to recognize L_3, and another to recognize L_5.

2) Write down RegEx's for L_3 and L_5, then for their union $L_3 \cup L_5$.

3) Construct the ε-NFA that recognizes $L_3 \cup L_5$.

4) Use the GNFA algorithm to obtain a RegEx for $L_3 \cup L_5$.

(4) Let $B_n = \{\mathtt{a}^m \mid m \text{ is a multiple of } n\} = \{\mathtt{a}^{kn} \mid k \in \mathbb{Z}_{\geq 0}\}$ over the alphabet $\Sigma = \{\mathtt{a}\}$.

Show that the language B_n is regular for any $n \in \mathbb{N}$ by writing a regular expression for it.

Outline the description of an NFA that can recognize it.

(5) **(Closure of regular languages under reversal of strings)**

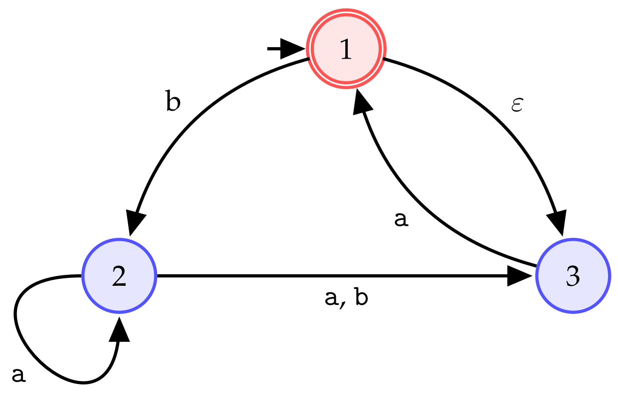
For any string $s = \mathtt{s}_1\mathtt{s}_2 \ldots \mathtt{s}_n$, where $\mathtt{s}_i$ are symbols from the alphabet, the **reverse** of s is the string s written in reverse order: $s^R = \mathtt{s}_n\mathtt{s}_{n-1} \ldots \mathtt{s}_1$.

Given an NFA or RegEx that recognizes a language A, describe how you can transform this NFA/RegEx to recognize the language $A^R = \{w^R \mid w \in A\}$, i.e. the language that contains all the strings from A but in the reverse order.

Hint: Test your ideas on the languages given by the RegEx's: ($\Sigma = \{\mathtt{a}, \mathtt{b}\}$)

$$\mathtt{a}, \mathtt{b}, \mathtt{aa}, \mathtt{ab}, \mathtt{aa} + \mathtt{bb}, \mathtt{ab} + \mathtt{bb}, \mathtt{a}^*\mathtt{b}^*, \Sigma^*\mathtt{a}, \mathtt{a}\Sigma^*, \mathtt{ab}^*\mathtt{a}^*\mathtt{b}, (\mathtt{ab})^*, (\mathtt{aa} + \mathtt{bb})^*, (\mathtt{ab} + \mathtt{bb})^*.$$

(1) Convert the following ε-NFA to an equivalent DFA.



(1) Show that

1) $1^*\emptyset = \emptyset$
(Concatenating the empty RegEx $\emptyset$ to any RegEx yields the empty RegEx again)

2) $\emptyset^* = \{\varepsilon\}$

You may find it helpful to construct the corresponding ε-NFAs.

(2) Let $\Sigma = \{0, 1\}$ and let

$$D = \{w \mid w \text{ contains an equal number of the occurrances of the substrings } 01 \text{ and } 10\}.$$

As an example, $101 \in D$ but $1010 \notin D$.
Show that D is a regular language (by producing an NFA for it, or otherwise).

Does this hold for $\{w \mid w$ contains an equal number of 0's and 1's$\}$?
Can you see why? What is the difference!?
How about the language $\{w \mid w$ contains a **non-equal** number of 0's and 1's$\}$?

(3) Extend your class for simulating DFAs and NFAs from the last lab to convert a given NFA into an equivalent DFA or to a RegEx.