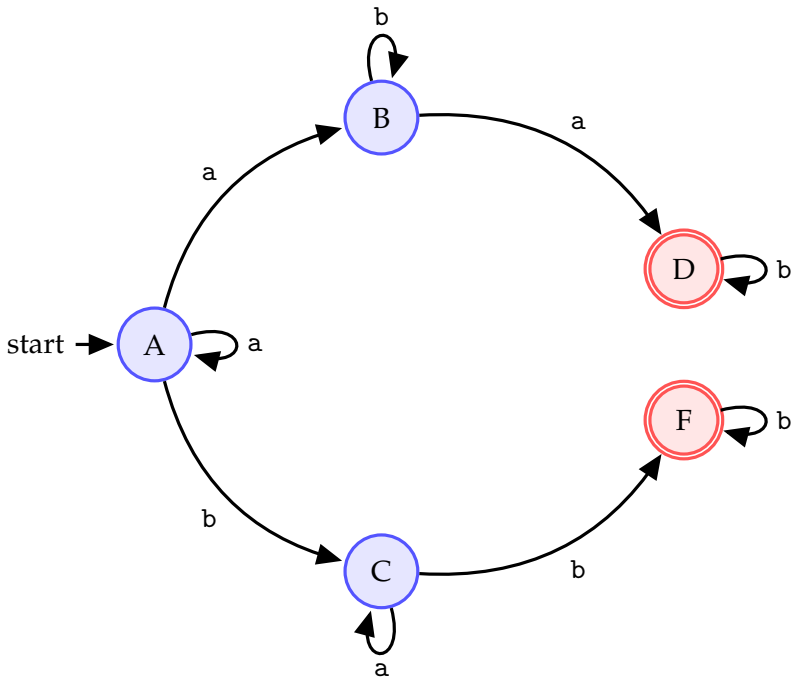
(1) Use the *subset construction method* to convert the following NFA to an equivalent DFA.



Give an informal description of the language recognized by this NFA/DFA.

Solution

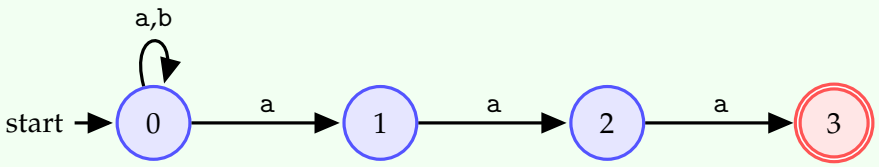
		a	b
→	{A}	{A,B}	{C}
	{A,B}	{A,B,D}	{B,C}
	{C}	{C}	{F}
*	{A,B,D}	{A,B,D}	{B,C,D}
	{B,C}	{C,D}	{B,F}
*	{F}	∅	{F}
*	{B,C,D}	{C,D}	{B,D,F}
*	{C,D}	{C}	{D,F}
*	{B,F}	{D}	{B,F}
	∅	∅	∅
*	{B,D,F}	{D}	{B,D,F}
*	{D,F}	∅	{D,F}
*	{D}	∅	{D}

Strings that start with zero or more a's followed by afollowed by zero or more b's followed by afollowed by zero or more b's; or strings that start with zero or more a's followed by bfollowed by zero or more a's followed by bfollowed by zero or more b's.

(English equivalent of the RegEx: $a^*ab^*ab^* + a^*ba^*bb^*$. Note that we can also write this as: $a^*(ab^*a + ba^*b)b^*$ – how would you informally read this?)

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

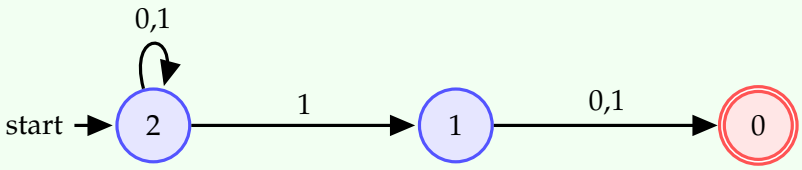
Solution



		a	b
→	{0}	{0,1}	{0}
	{0,1}	{0,1,2}	{0}
	{0,1,2}	{0,1,2,3}	{0}
*	{0,1,2,3}	{0,1,2,3}	{0}

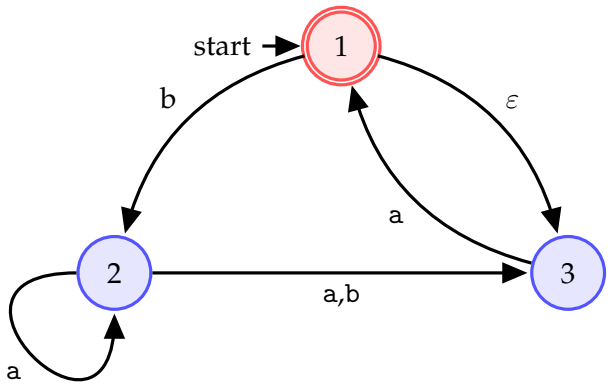
(3) Design an NFA that accepts strings over $\{\texttt{0}, \texttt{1}\}$ which have `1` in the second position from the end (e.g. `0010`, `1011`,`10`, etc.), then convert it to an equivalent DFA.

Solution



		0	1
→	{2}	{2}	{2,1}
	{2,1}	{2,0}	{2,1,0}
*	{2,0}	{2}	{2,1}
*	{2,1,0}	{2,0}	{2,1,0}

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



Solution

		aε*	bε*
→*	{1}	∅	{2}
	∅	∅	∅
	{2}	{2,3}	{3}
	{2,3}	{1,2,3}	{3}
	{3}	{1,3}	∅
*	{1,2,3}	{1,2,3}	{2,3}
*	{1,3}	∅	{2}

or

		ε*a	ε*b
→*	{1,3}	{1,3}	{2}
	{2}	{2,3}	{3}
	{2,3}	{1,2,3}	{3}
	{3}	{1,3}	∅
*	{1,2,3}	{1,2,3}	{2,3}
	∅	∅	∅

These are equivalent. (In the first transition table: the first and last rows describe the same "behaviour," so we can simplify it by keeping only one of the two rows.)

(1) **(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^*	Strings that start with @
`@.*|.*@`	`@`Σ^* + Σ^*`@`	Strings that either start or end with @
`@.*@`	`@`Σ^*`@`	Strings that either start and end with @
`@=.*=@`	`@=`Σ^*`=@`	Strings that start with @= and end with =@

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 gray 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*.

a) $01 + 10 = \{\boxed{01}, \boxed{10}\}$

b) $(\varepsilon + 0)(\varepsilon + 1) = \{\varepsilon, 0, 1, \boxed{01}\}$

c) $(0 + \varepsilon)1^* = 01^* + 1^* = \{w \mid w$ has at most $\boxed{\text{one zero}}$ and is at the start of $w\}$

d) $\Sigma^*0 = \{w \mid w$ ends with a $\boxed{0}\} = \{w \mid w$ respresents an $\boxed{\text{even}}$ number in binary$\}$

e) $0^*10^* = \{w \mid w$ contains a single $\boxed{1}\}$

f) $\Sigma^*0\Sigma^* = \{w \mid w$ has at least one $\boxed{0}\}$

g) $\Sigma^*001\Sigma^* = \{w \mid w$ contains the string $\boxed{001}$ as a substring$\}$

h) $\Sigma^*000^*\Sigma^* = \{w \mid w$ cotains at least $\boxed{2}$ consective $\boxed{0}$'s$\}$

i) $(011^*)^* = \{w \mid$ every $\boxed{0}$ in w is followed by at least one $\boxed{1}\}$

j) $\Sigma\Sigma + \Sigma\Sigma\Sigma = \Sigma\Sigma(\varepsilon + \Sigma) = \{w \mid$ the length of w is exactly $\boxed{2}$ or $\boxed{3}\}$

k) $(\Sigma\Sigma)^* = \{w \mid w$ is a string of $\boxed{\text{even}}$ length$\}$

l) $(\Sigma\Sigma\Sigma)^* = \{w \mid$ the length of w is a multiple of $\boxed{3}\}$

m) $0\Sigma^*0 + 1\Sigma^*1 + 0 + 1 = \{w \mid w$ starts and ends with the $\boxed{\text{same}}$ symbol$\}$

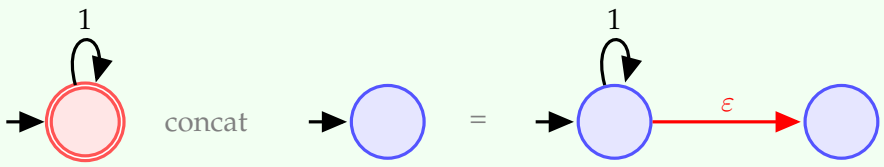
n) $1^*\emptyset = \emptyset$ ...why!?
(Concatenating the empty RegEx $\emptyset$ to any RegEx yields the empty RegEx again)

o) $\emptyset^* = \{\varepsilon\}$ ...why!?

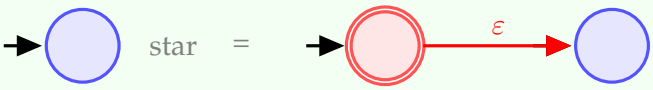
For the last two, you may find it helpful to construct the corresponding ε-NFAs.

Solution

n)



o)



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

a) The language $L_{\mathtt{a}}$ of all strings that start with $\mathtt{a}$. $\boxed{\mathtt{a}\Sigma^*}$

b) The language $L_{\mathtt{b}}$ of all strings that end with $\mathtt{b}$. $\boxed{\Sigma^*\mathtt{b}}$

c) The union $L_{\mathtt{a}} \cup L_{\mathtt{b}}$. $\boxed{\mathtt{a}\Sigma^* + \Sigma^*\mathtt{b}}$

d) The concatenation $L_{\mathtt{a}}L_{\mathtt{b}}$. $\boxed{\mathtt{a}\Sigma^*\Sigma^*\mathtt{b}} = \boxed{\mathtt{a}\Sigma^*\mathtt{b}}$

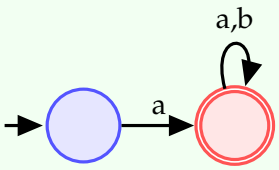
e) $L = (L_{\mathtt{a}} \cup L_{\mathtt{b}})L_{\mathtt{a}}L_{\mathtt{b}}$. $\boxed{(\mathtt{a}\Sigma^* + \Sigma^*\mathtt{b})\mathtt{a}\Sigma^*\mathtt{b}}$

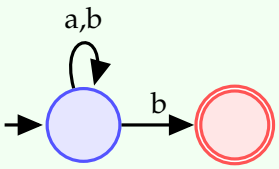
f) The closure of L: L^*. $\boxed{\Big((\mathtt{a}\Sigma^* + \Sigma^*\mathtt{b})\mathtt{a}\Sigma^*\mathtt{b}\Big)^*}$

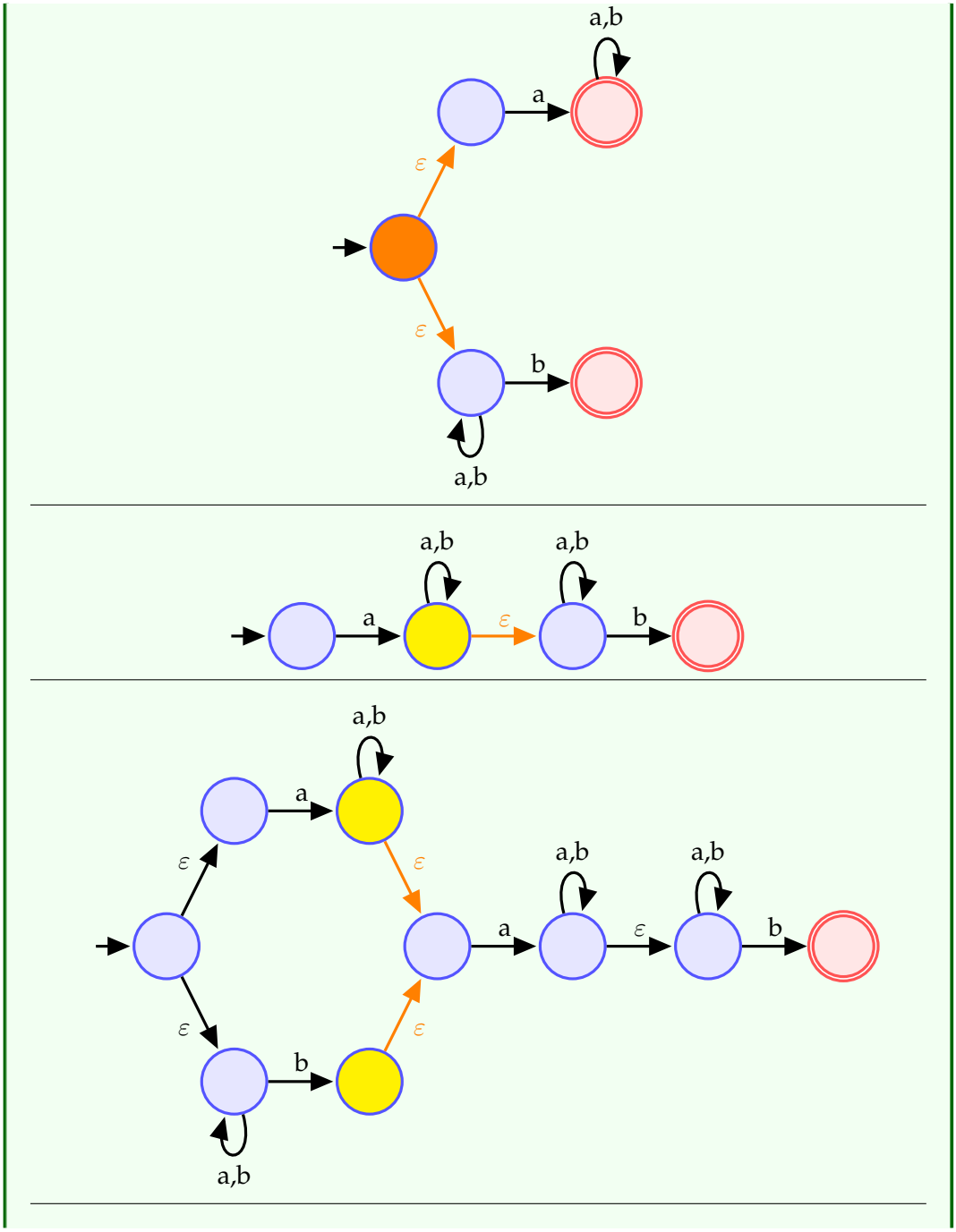
Produce ε-NFAs for each of the above using the constructions shown in the lecture for the union, concatenation, and star.

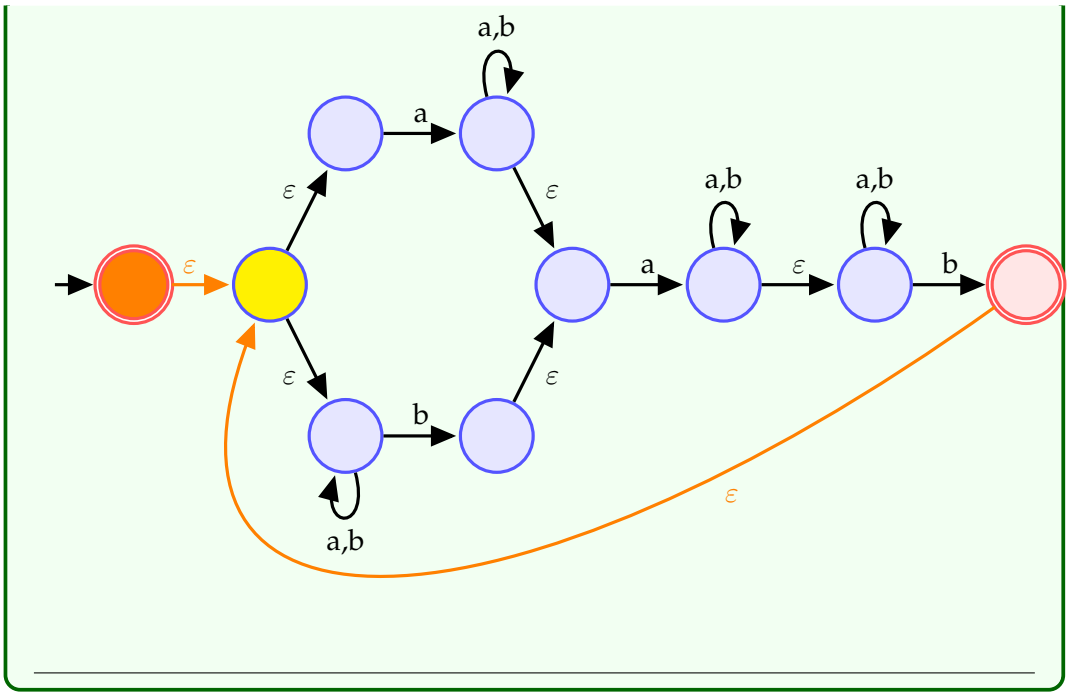
You can refer to the last page for the graphical illustrations for these constructions.

Solution






a,b
a
ε
ε
b
a,b
a,b
a,b
a
ε
b
a,b
a,b
a
ε
a,b
a,b
a
ε
ε
b
a,b



(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.

a) $\mathtt{a}^*\mathtt{b}^*$ e.g. a, b and ba, aba

b) $\mathtt{a}(\mathtt{ba})^*\mathtt{b}$ e.g. ab, abab and aab, aabb

c) $\mathtt{a}^* + \mathtt{b}^*$ e.g. ϵ, a and ab, ba

d) $(\mathtt{aaa})^*$ e.g. ϵ, aaa and a, aa

e) $\Sigma^*\mathtt{a}\Sigma^*\mathtt{b}\Sigma^*\mathtt{a}\Sigma^*$ e.g. aba, aaba and a, ab

f) $\mathtt{aba} + \mathtt{bab}$ e.g. aba, bab and a, ab

g) $(\varepsilon + \mathtt{a})\mathtt{b}$ $= \mathtt{b} + \mathtt{ab}.$ e.g. b, ab and a, ba

h) $(\mathtt{a} + \mathtt{ba} + \mathtt{bb})\Sigma^*$ e.g. a, ba and ϵ, b

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

a) $\{w \mid w$ begins with 1 and ends with a $0\}$ $1\Sigma^*0$

b) $\{w \mid w$ contains at least three 1's$\}$ $\Sigma^*1\Sigma^*1\Sigma^*1\Sigma^*$

c) $\{w \mid w$ contains the substring $0101\}$ $\Sigma^*0101\Sigma^*$

d) $\{w \mid w$ has length at least 3 and its third symbol is $0\}$ $\Sigma\Sigma0\Sigma^*$

e) $\{w \mid w$ starts with 0 and has odd length, **or** starts with 1 and has even length$\}$ $0(\Sigma\Sigma)^* + 1\Sigma(\Sigma\Sigma)^*$ $= (0 + 1\Sigma)(\Sigma\Sigma)^*$

f) $\{w \mid w$ does not contain the substring $110\}$ $(0 + (10)^*)^*1^*$

g) $\{w \mid$ the length of w is at most $5\}$ $\varepsilon + \Sigma + \Sigma\Sigma + \Sigma\Sigma\Sigma + \Sigma^4 + \Sigma^5$

h) $\{w \mid w$ is any string except 11 and 111$\}$
$\boxed{\varepsilon + 1 + 1111\Sigma^* + \Sigma^*0\Sigma^*}$ or $\boxed{(0 + 10 + 110 + 1111^*0)^*(11 + 1111^*)}$

i) $\{w \mid$ every odd position of w is 1$\}$ $\boxed{(\Sigma 1)^*}$

j) $\{w \mid w$ contains at least two 0's and at most one 1$\}$ $\boxed{000^* + 000^*10^* + 0100^* + 1000^*}$
= $\boxed{0^*(001 + 010 + 100)0^*}$

k) $\{\varepsilon, 0\}$ $\boxed{\varepsilon + 0}$

l) $\{w \mid w$ contains an even number of 0's, or contains exactly two 1's$\}$
$\boxed{(1^*01^*01^*)^* + 0^*10^*10^*}$

m) The empty set. $\boxed{\emptyset}$

n) All strings except the empty string. $\boxed{\Sigma\Sigma^*}$ also written as $\boxed{\Sigma^+}$. (i.e. strings of length ≥ 1)

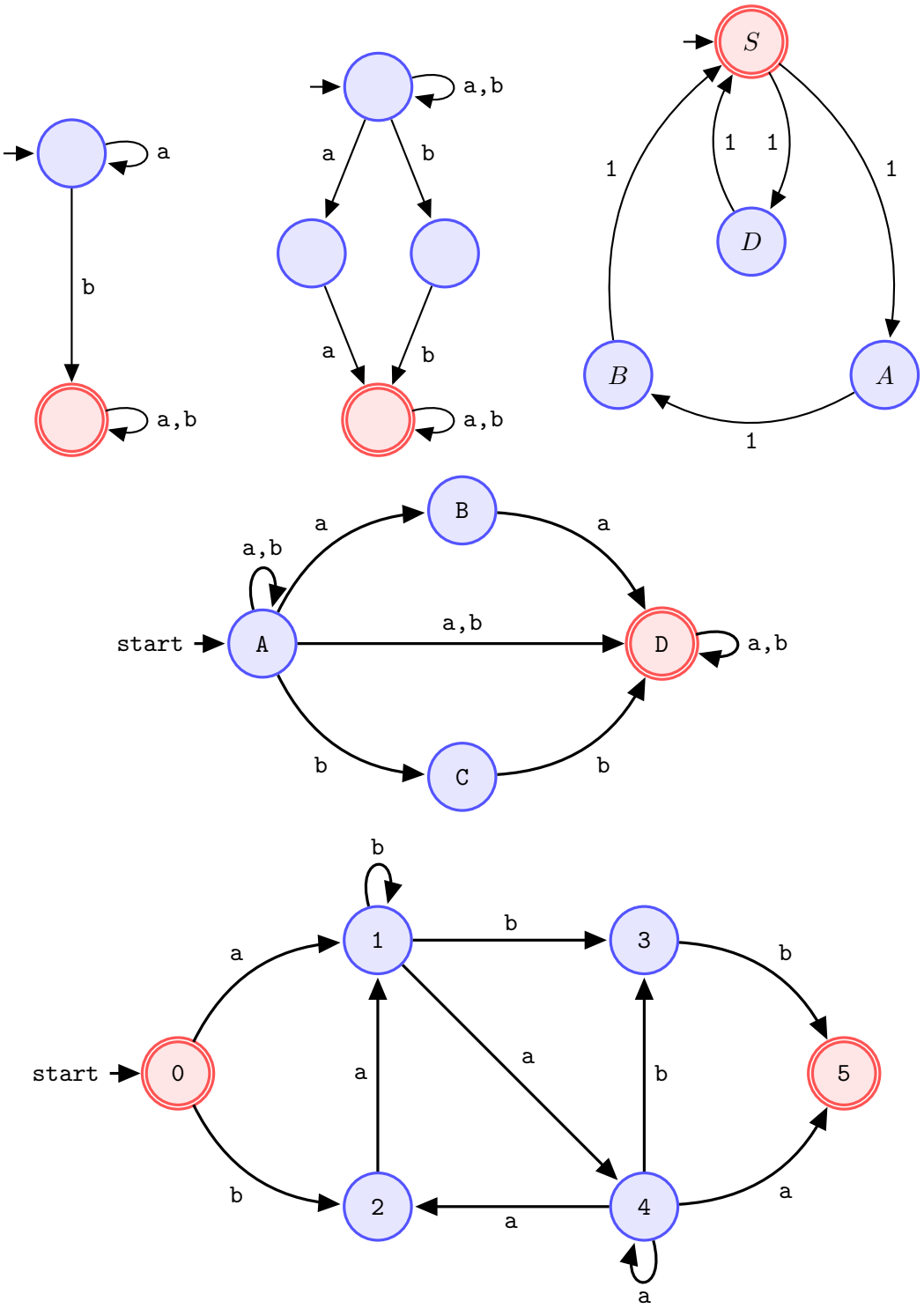
Reminder: We can convert any NFA into a GNFA as follows:

- 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.

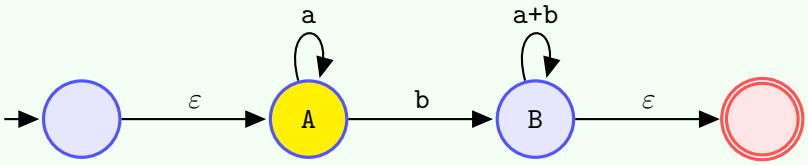
(1) Use the GNFA algorithm to find regular expressions for the languages recognized by the following NFAs.

Can you interpret the results?

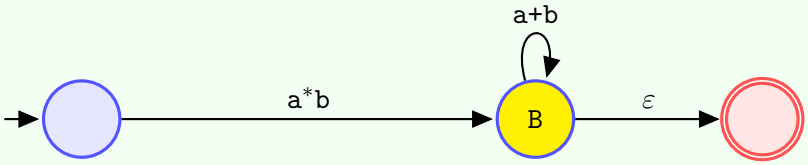


Solution

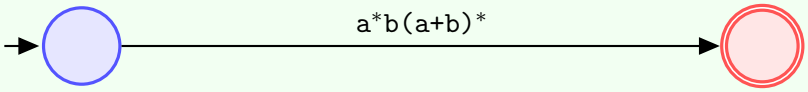
Convert to GNFA:



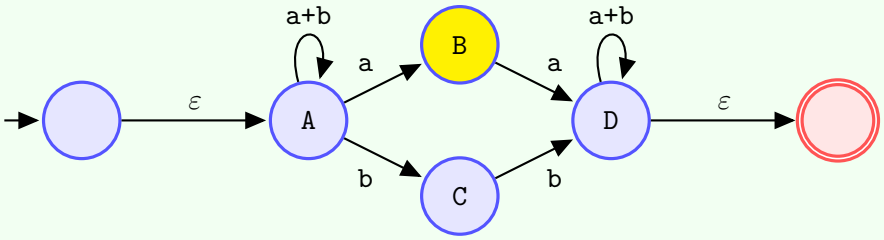
Remove A:



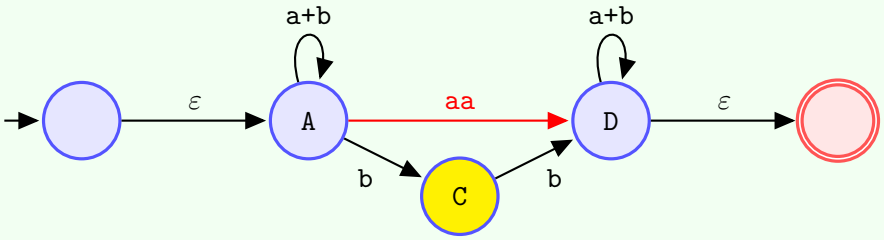
Remove B:



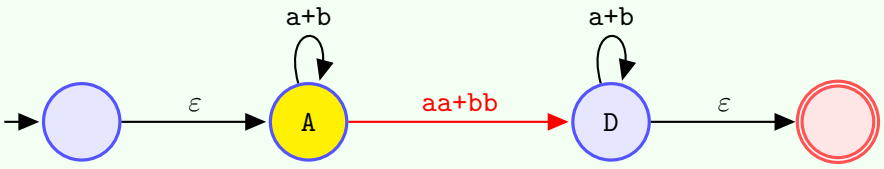
Convert to GNFA:



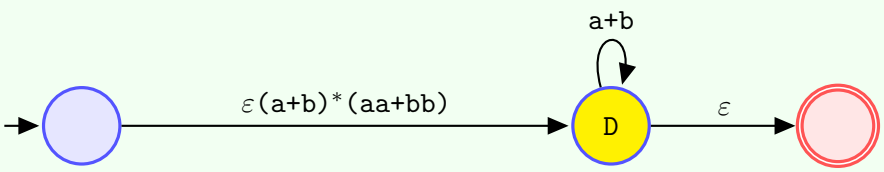
Remove B:



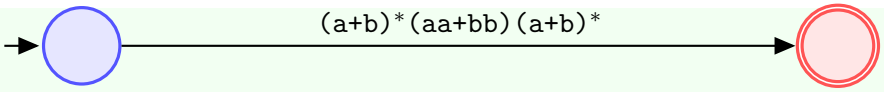
Remove C:



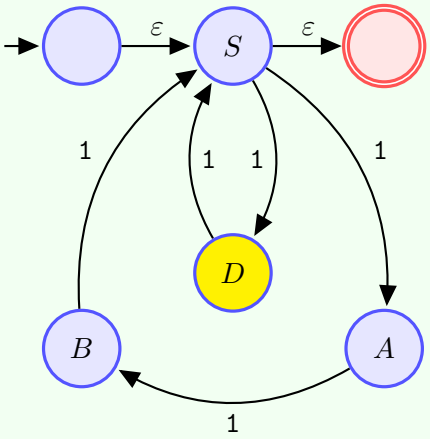
Remove A:



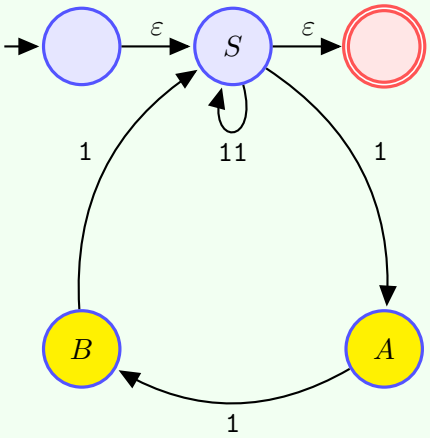
Remove D:



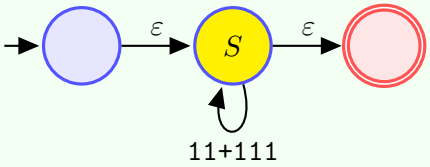
Convert to GNFA:



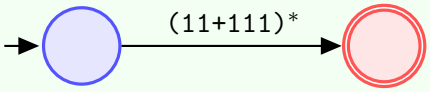
Remove D:



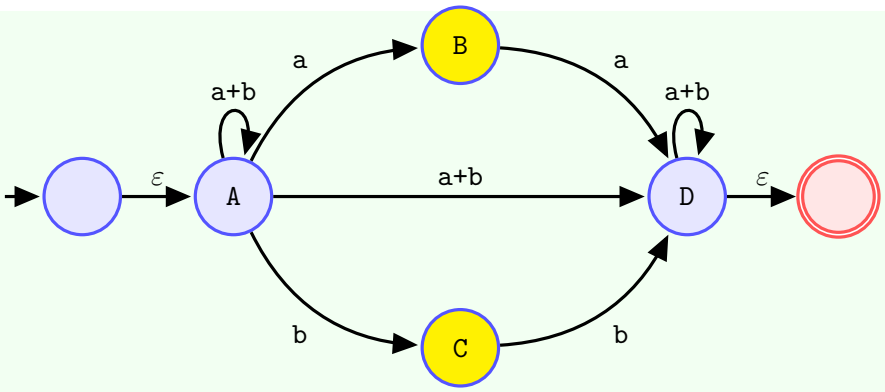
Remove A then D in succession:



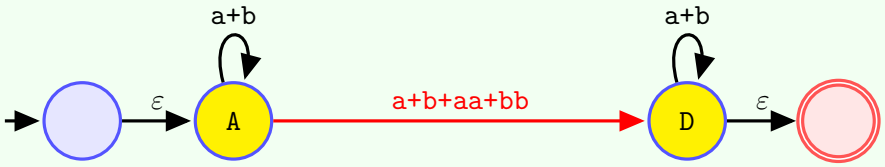
Remove S:



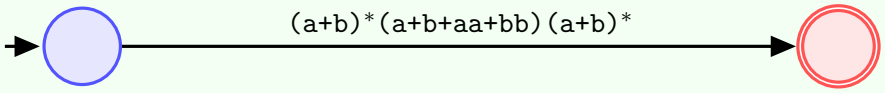
Convert to GNFA:



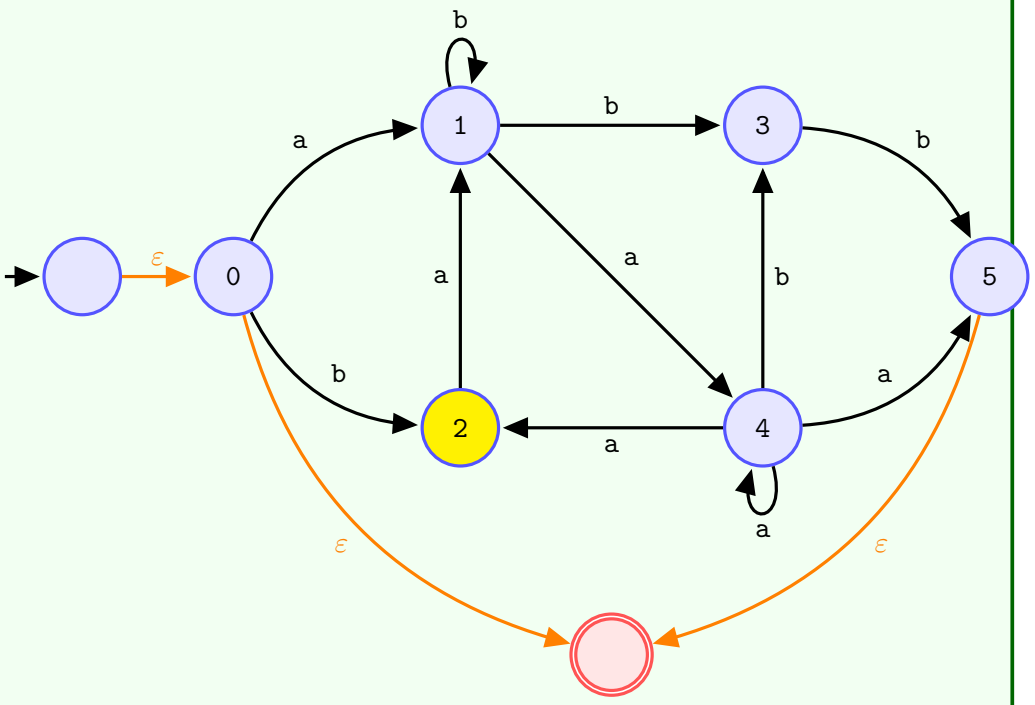
Remove B then C in succession:



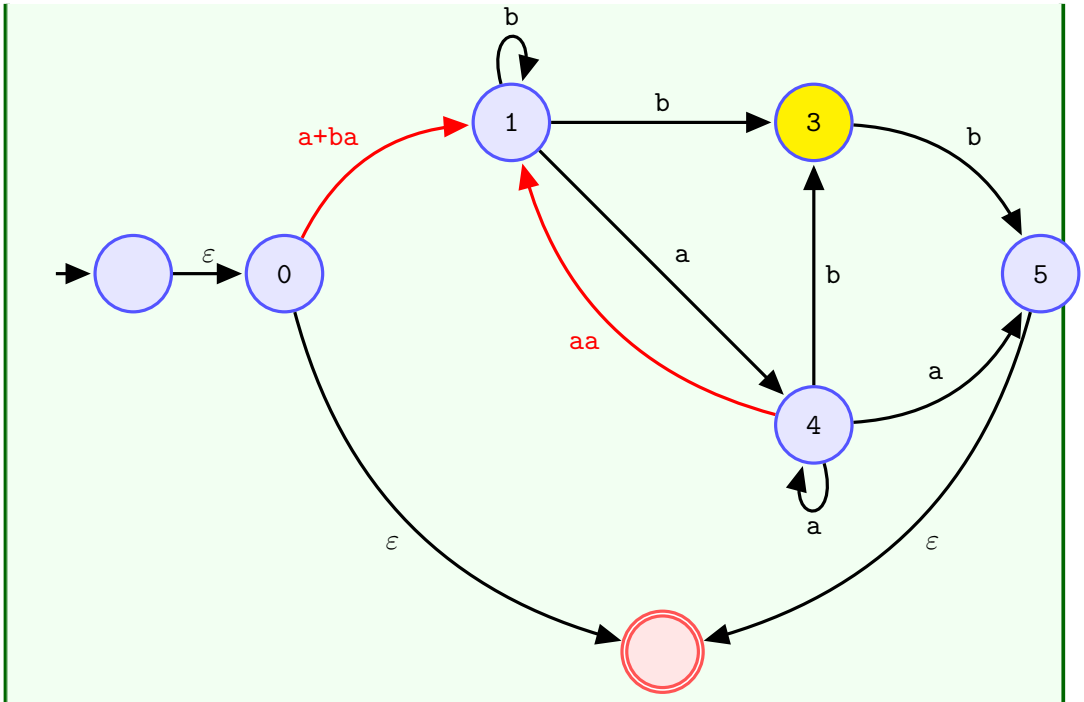
Remove A then D in succession:



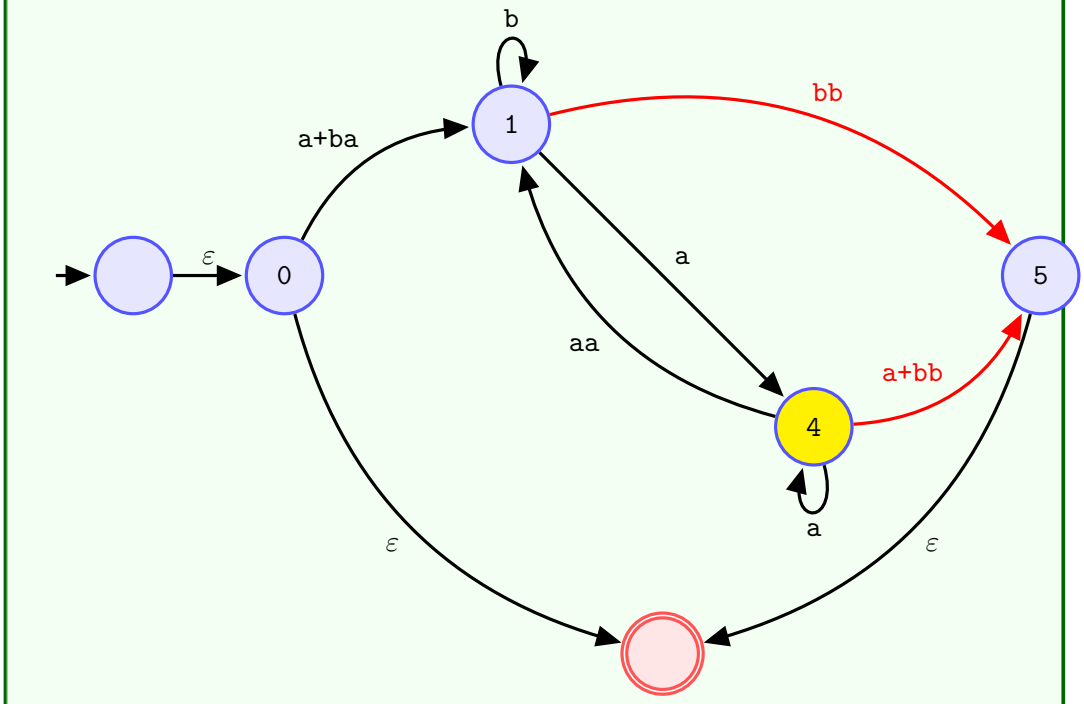
Convert to GNFA:



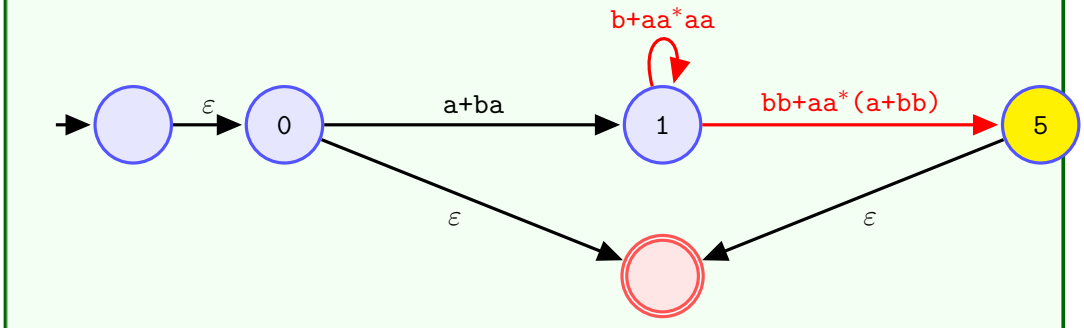
Remove 2:



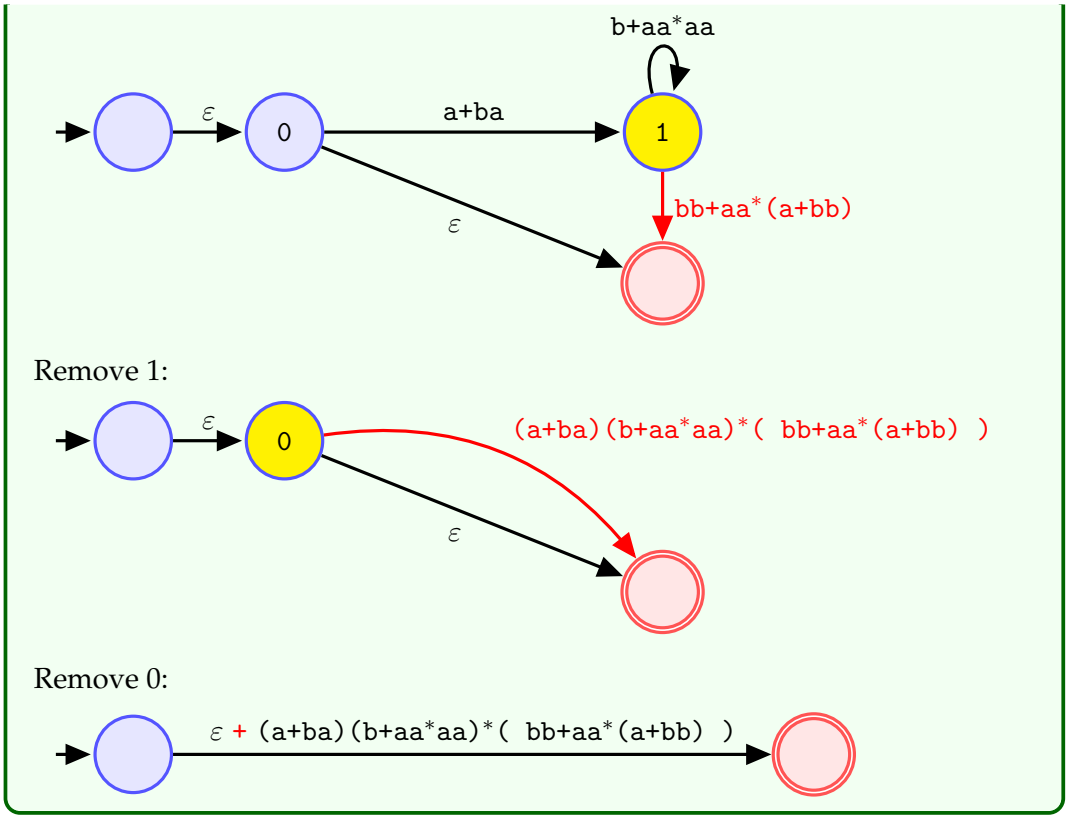
Remove 3:



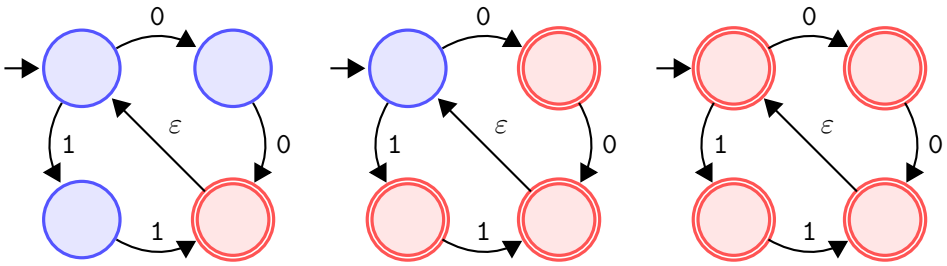
Remove 4:



Remove 4:



(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. $n \in \mathbb{N} = \{1, 2, 3, \ldots\}$).

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

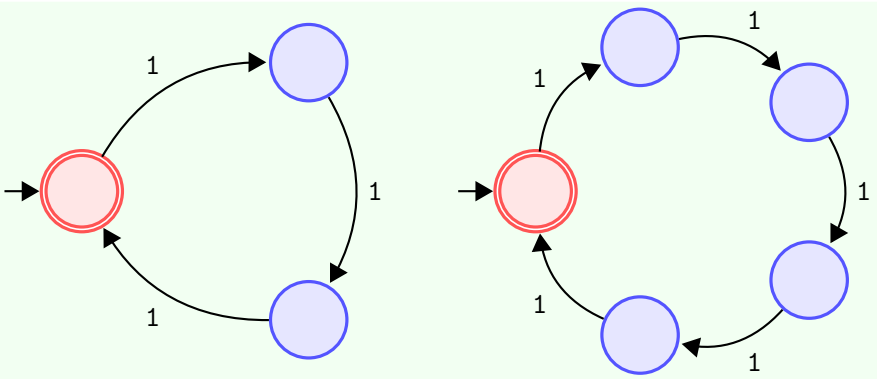
b) Write down RegEx's for L_3 and L_5, then for their union $L_3 \cup L_5$.

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

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

Solution

a)

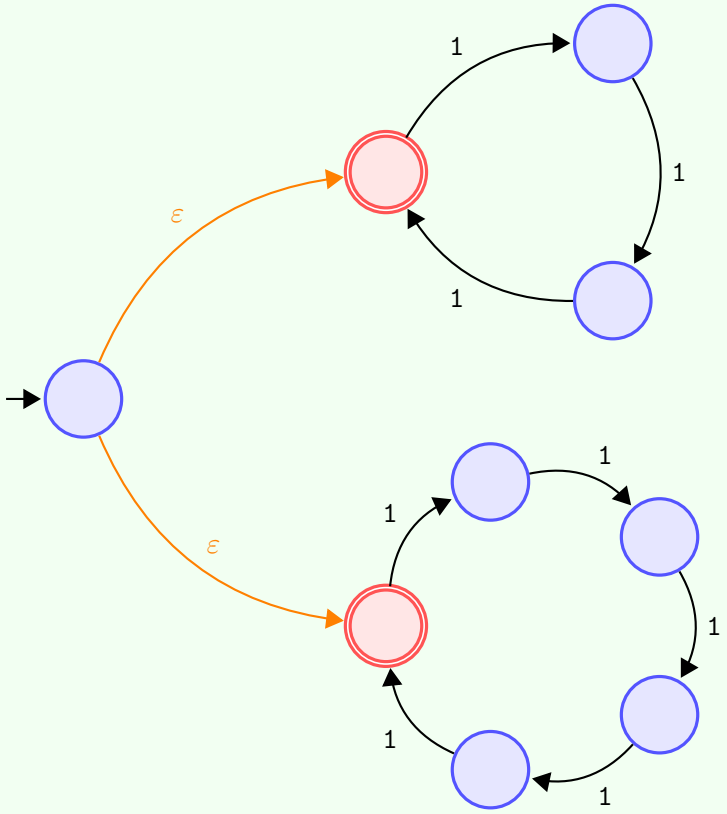


b) L_3: $(\mathtt{111})^* = (1^3)^*$

L_5: $(\mathtt{11111})^* = (1^5)^*$

$L_3 \cup L_5$: $(1^3)^* + (1^5)^*$.

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



d) 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.

Solution

RegEx: $(\mathtt{a}^k)^*$

NFA is a generalization of the case for L_3 and L_5 above. It would have k states $q_0, \ldots, q_{k-1}$ with q_0 being the initial state, which is also the only accepting state.

transitions go on 1 from q_i to q_{i+1} for $i = 0, \ldots, k-2$ and from q_{k-1} to q_0.

(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.

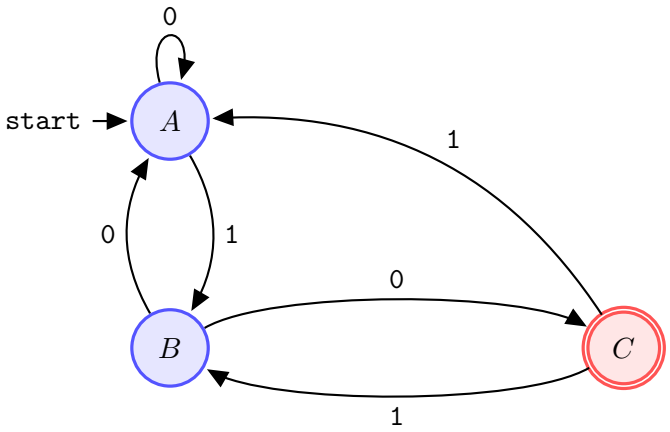
Hint: Test your ideas on the languages given by the RegEx's: ($\Sigma = \{\mathtt{a}, \mathtt{b}\}$)

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

Solution

Basic idea: reverse the arrows in the state diagram, but need to address the case with many accepting states... etc.

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



- 10011010
- 01010011
- 00010111
- 0010010

Now, convert the above NFA into a DFA.

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

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

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

$D = \{w \mid w$ contains an equal number of the occurances of the substrings 01 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) 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 $\mathcal{M}_1 = (Q_1, \Sigma, \delta_1, q_{0,1}, F_1)$ and $\mathcal{M}_2 = (Q_2, \Sigma, \delta_2, q_{0,2}, F_2)$ are DFAs recognizing L_1 and L_2, respectively.
Let $\mathcal{M}$ be the DFA given by $(Q, \Sigma, \delta, q_0, F)$ where

$$\begin{aligned} Q &= Q_1 \times Q_2 \\ q_0 &= (q_{0,1}, q_{0,2}) \end{aligned}$$

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

a) Let the languages L_1 and L_2 be given by the two RegEx's: $\texttt{b}^*\texttt{a}\Sigma^*$ and $\texttt{a}^*\texttt{b}\Sigma^*$, respectively, where $\Sigma = \{\texttt{a}, \texttt{b}\}$.

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

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

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

b) Use the same method for the language

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

c) **(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 $L(\Sigma^*) - L_2 = L_1 - L_2$. 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. (Counterexample)

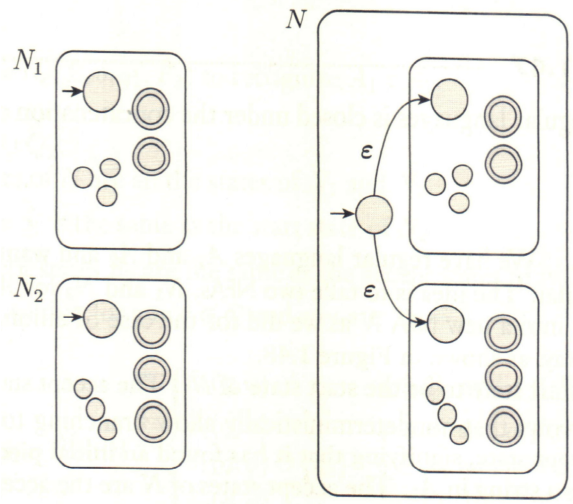
Solution

To be completed.

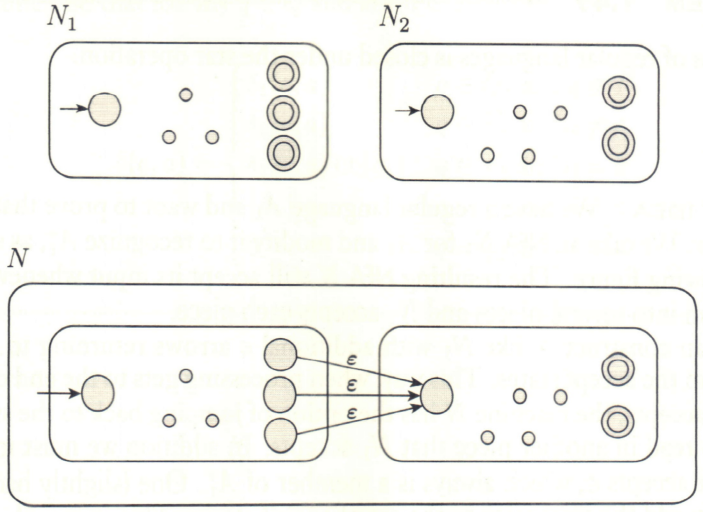
First create a DFA that accepts the strings that do **not** satisfy the required property, then flip the accepting states into non-accepting ones, and vice versa.

This will produce a DFA that accepts the required strings.

Union:



Concatenation:



Star: