1

Exercises

Exercise n°1

Find a grammar for $\Sigma = \{a, b\}$ that generates the set of all strings with no more than three a's.

Exercise n°2

What language does the grammar with these productions generate?

$$S \to Aa$$
$$A \to B$$
$$B \to Aa$$

Exercise n°3

Let $\Sigma = \{a, b\}$. For each of the following languages, find a grammar that generates it.

- $L_1 = \{a^n b^m \mid n \geq 0, m > n\}$
- $L_2 = \{a^n b^{2n} \mid n \geq 0\}$
- $L_3 = \{a^{n+2} b^n \mid n \geq 1\}$
- $L_4 = \{a^n b^{n-3} \mid n \geq 3\}$

Exercise n°4

Find a regular expression for the set $\{a^n b^m : (n+m) \text{ is even}\}$.

Exercise n°5

What languages do the expressions $(\emptyset^*)^*$ and $a\emptyset$ denote?

Exercise n°6

Find a regular expression for

$$L = \{vwv \mid v, w \in \Sigma^* \wedge |v| = 2\} \text{ where } \Sigma = \{a, b\}$$

Exercise n°7

Give regular expressions for the following languages on $\Sigma = \{a, b, c\}$.

- all strings containing exactly one a,
- all strings containing no more than three a's,
- all strings that contain at least one occurrence of each symbol in Σ
- all strings that contain no run of a's of length greater than two,
- all strings in which all runs of a's have lengths that are multiples of three.

Exercise n°8

Find DFA's that accept the following languages.

- $L(ab(a+ab)^*(a+aa))$
- $L(((aa^*)^*b)^*)$

Exercise n°9

Construct a DFA that accepts the language generated by the grammar

$$
\begin{aligned}
S &\to abA \\
A &\to baB \\
B &\to aA \mid bb
\end{aligned}
$$

Exercise n°10

Construct a left-linear grammar for the language generated by the grammar

$$
\begin{aligned}
S &\to abA \\
A &\to baB \\
B &\to aA \mid bb
\end{aligned}
$$

Exercise n°11

Construct a right-linear grammar for the language $L((aab^*ab)^*)$.

Exercise n°12

Find a regular grammar that generates the language

$$L = \{w \in \{a,b\}^* : n_a(w) + 3n_b(w) \text{ is even}\}$$

2

solution

Solution n°1

$$
\begin{aligned}
S &\to B \mid BaB \mid BaBaB \mid BaBaBaB \\
B &\to \lambda \mid bB
\end{aligned}
$$

Solution n°2

This grammar generates the empty set (no word can be derived) since there is no finite derivation of a terminal word.

Solution n°3

$L_1 = \{a^n b^m \mid n \geq 0, m > n\}$:

$$\begin{aligned} S &\to Ab \\ A &\to \lambda \mid Ab \mid aAb \end{aligned} \qquad \text{or} \qquad S \to aSb \mid Sb \mid b$$

$L_2 = \{a^n b^{2n} \mid n \geq 0\}$:

$$S \to \lambda \mid aSbb$$

$L_3 = \{a^{n+2} b^n \mid n \geq 1\}$:

$$\begin{aligned} S &\to aaaBb \\ B &\to aBb \mid \lambda \end{aligned} \qquad \text{or} \qquad S \to aSb \mid aaab$$

$L_4 = \{a^n b^{n-3} \mid n \geq 3\}$:

$$\begin{aligned} S &\to aaaA \\ A &\to aAb \mid \lambda \end{aligned} \qquad \text{or} \qquad S \to aaa \mid aSb$$

Solution n°4

First method:

Define $E = 2\mathbb{N}$ the set of even and positive integers and $O = \mathbb{N} \setminus E$ the set of odd and positive integers.

$$m + n \text{ is even} \Longleftrightarrow (m,n) \in E \times E \quad \vee \quad (m,n) \in O \times O$$

Therefore we can rewrite $\{a^n b^m : (n+m) \text{ is even}\}$ as :

$$\{a^n b^m : (m,n) \in E \times E\} \cup \{a^n b^m : (m,n) \in O \times O\}$$

A regular expression for the set $\{a^n b^m : (m,n) \in E \times E\}$ is $(aa)^*(bb)^*$.
A regular expression for the set $\{a^n b^m : (m,n) \in O \times O\}$ is $a(aa)^*b(bb)^*$.

Therefore $(aa)^*(bb)^* + a(aa)^*b(bb)^*$ is suitable.

Second method:

We can easily construct an NFA and convert it into a regEX. This one is easy, we can directly guess: $(aa)^*(\lambda + (ab+bb)(bb)^*\lambda) = (aa)^*(\lambda + ab)(bb)^*$

Solution n°5

The n-th power of a language L is defined by induction on n:

$$\mathrm{L}^0 = \{\lambda\} \quad \mathrm{L}^{n+1} = L^n L$$

Every regular expression r defines a language $L(r)$ hence $\emptyset = L(\emptyset)$.

$$\begin{aligned}
(\emptyset^*)^* &= (L(\emptyset)^*)^* \\
&= (\{\}^*)^* \\
&= \left(\{\}^0 \cup \{\}^1 \cup \{\}^2 \cup \cdots\right)^* \\
&= (\{\lambda\} \cup \{\} \cup \{\}\{\} \cup \cdots)^* \\
&= (\{\lambda\})^* \\
&= \{\lambda\}^0 \cup \{\lambda\}^1 \cup \{\lambda\}^2 \cup \cdots \\
&= \{\lambda\} \cup \{\lambda\} \cup \{\lambda\} \cup \cdots \\
&= \{\lambda\}
\end{aligned}$$

We could have used $L(r)^* = L(r^*)$ where $r = \lambda$ and conclude with $\lambda^* = \lambda$.

$$\begin{aligned}
a\emptyset &= \{a\}\{\} \\
&= \{\}
\end{aligned}$$

Notice that the concatenation of the sets $\{a\}$ and $\{\}$ consists of the words that can be created by concatenating a word from the set $\{a\}$ with a words from the set $\{\}$. Since there is no word in $\{\}$, there is no word in $\{a\} \cdot \{\}$.

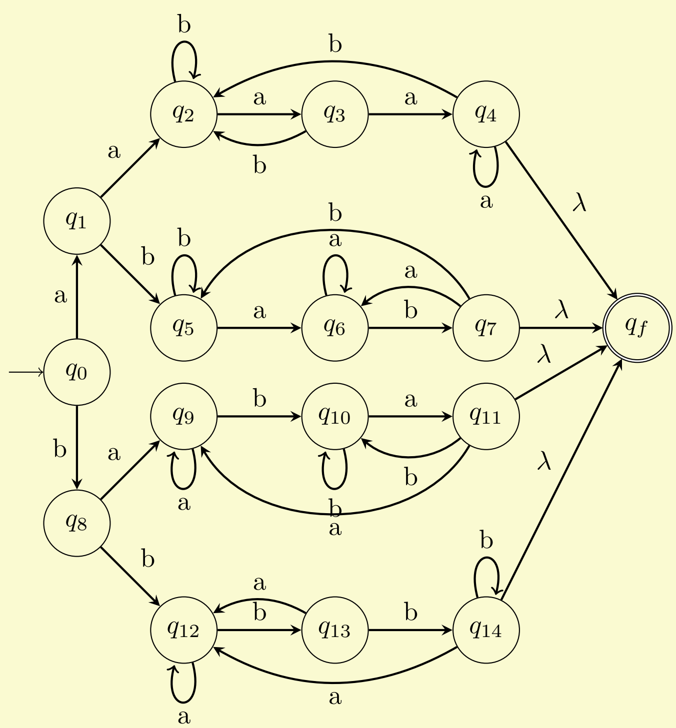
Solution n°6

Method 1 Guess directly:

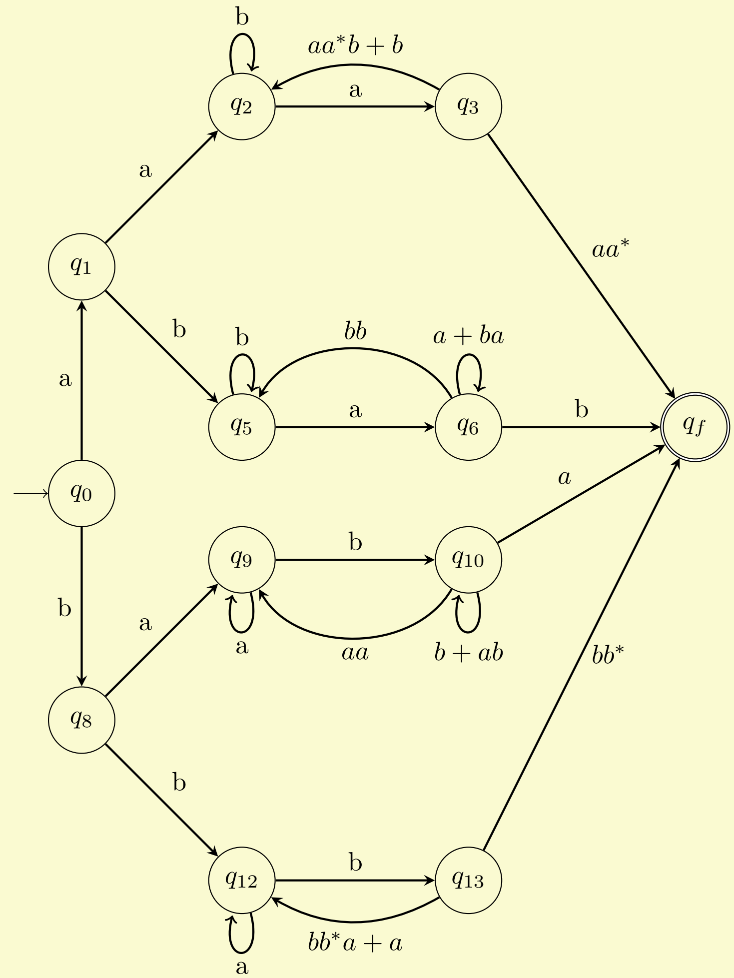
$$(aa(a+b)^*aa) + (ab(a+b)^*ab) + (ba(a+b)^*ba) + (bb(a+b)^*bb)$$

Method 2 We already have construct an DFA for this language in the exercise 2.1.11 so we only had to convert it in a regex.

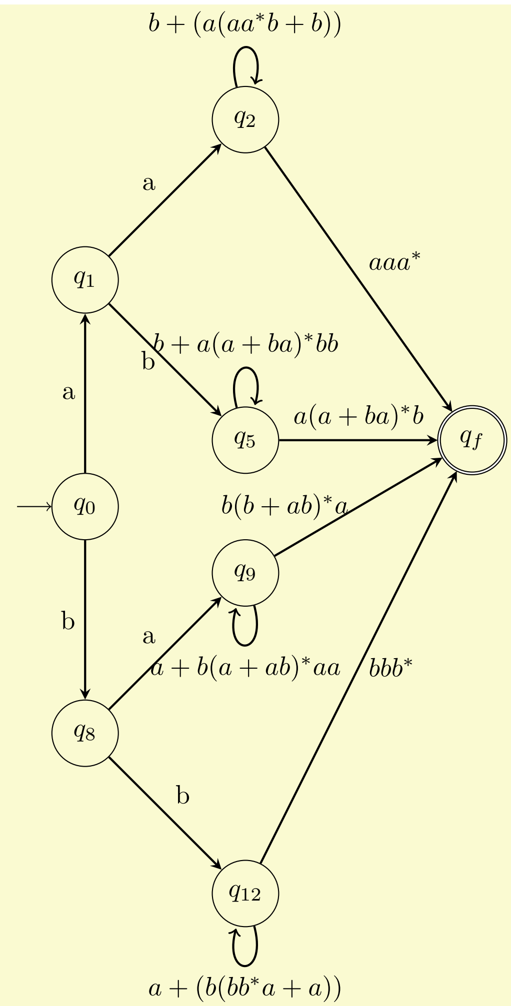
We transform our automaton such that there is one initial and one final state:



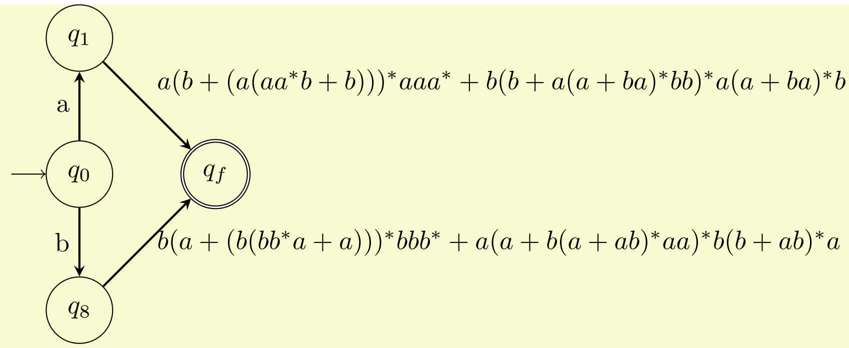
We delete q_4, q_7, q_{11}, q_{14} and merge all double arrows:



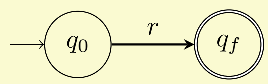
We delete q_3, q_6, q_{10}, q_{13} and merge all double arrows:



We delete q_2, q_5, q_9, q_{12} and merge all double arrows:



We delete q_1, q_8 and merge all double arrows:



$$\begin{aligned} r := {} & a(a(b+(a(aa^*b+b)))^*aaa^*+b(b+a(a+ba)^*bb)^*a(a+ba)^*b) \\ & + b(b(a+(b(bb^*a+a)))^*bbb^*+a(a+b(a+ab)^*aa)^*b(b+ab)^*a) \end{aligned}$$

Solution n°7

- all strings containing exactly one a:

$$(b+c)^*a(b+c)^*$$

- : all strings containing no more than three a's:

$$\begin{aligned}(b+c)^* &+ (b+c)^*a(b+c)^* \\ &+ (b+c)^*a(b+c)^*a(b+c)^* \\ &+ (b+c)^*a(b+c)^*a(b+c)^*a(b+c)^*\end{aligned}$$

- all strings that contain at least one occurrence of each symbol in:

  Let $Z := (a+b+c)^*$ as

$$Z(aZbZc + aZcb + bZcZa + bZaZc + cZaZb + cZbZa)Z$$

- all strings that contain no run of a's of length greater than two

$$((b+c)^* + (b+c)^*a(b+c)^* + (b+c)^*aa(b+c)^*)^*$$

  Equivalently,

$$(b+c+a(b+c)+aa(b+c))^*(\lambda + a + aa)$$

- all strings in which all runs of a's have lengths that are multiples of three:

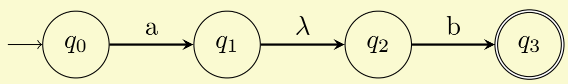
$$(b+c+aaa)^*$$

Solution n°8

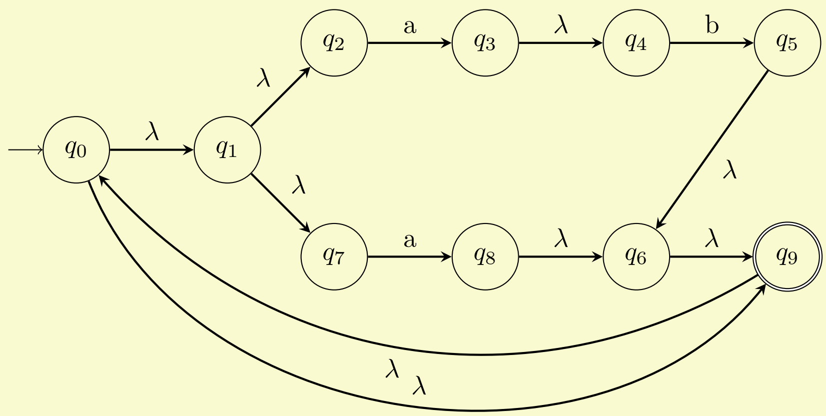
$$ab(a+ab)^*(a+aa)$$

:

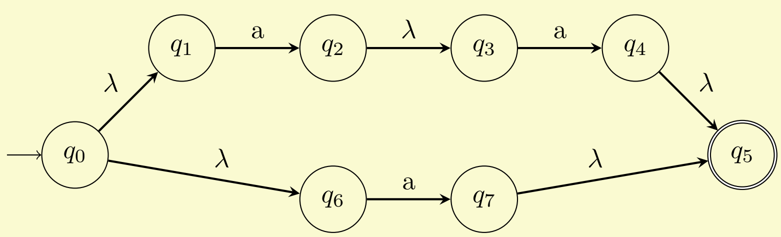
We create ab:



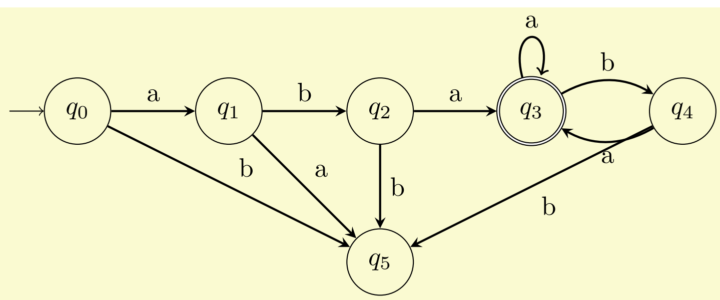
We create $(a+ab)^*$:



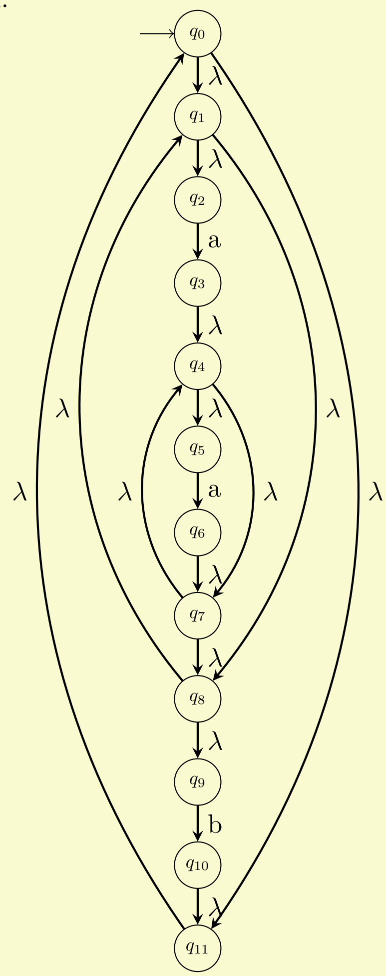
We create a+aa

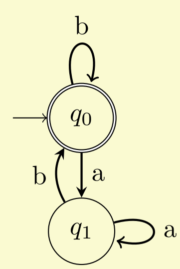


We can merge those tree NFA to obtain an NFA which is converted into DFA:



We create a NFA corresponding to $((aa^*)^*b)^*$ and we convert it into a DFA:





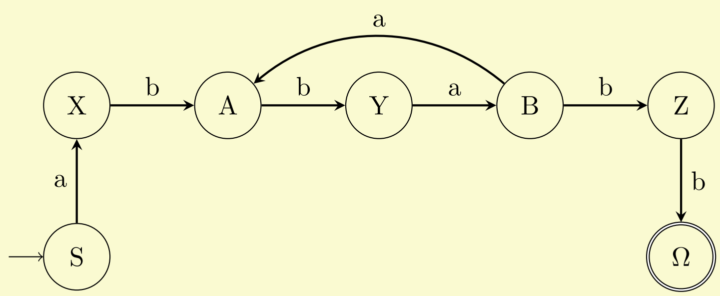
Solution n°9

First we need to transform the following grammar into a strictly right-linear grammar:

$$S \to abA$$
$$A \to baB$$
$$B \to aA \mid bb$$

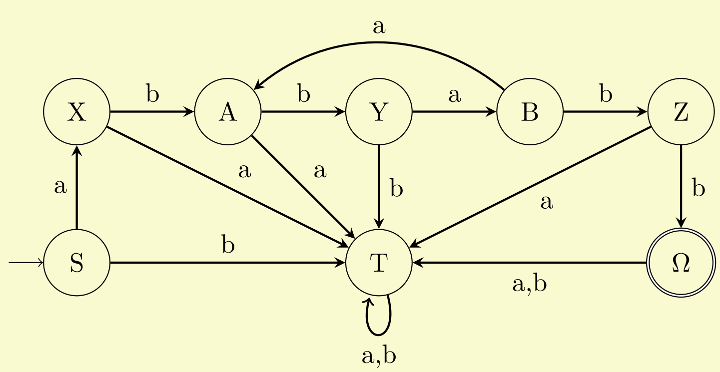
$$S \to aX$$
$$X \to bA$$
$$A \to bY$$
$$Y \to aB$$
$$B \to aA \mid bZ$$
$$Z \to b$$

The states of our automaton are the variables of the grammar plus an addition state for acceptance:



Now we create a DFA from our NFA. We can use the algorithm to do this, or we can observe that there are no λ-transitions and no state has two outgoing transitions with the same letter.So the NFA is already deterministic, but not yet complete.

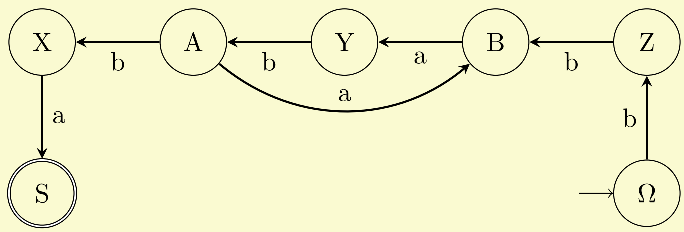
In deed, a DFA needs exactly one transition per letter, per state. In this case, we can simply introduce a 'trap' state T and direct the missing transitions to it:



Solution n°10

$$\begin{aligned} L \text{ is regular} &\iff L^R \text{ is regular} \\ &\iff \text{right linear grammar for } L^R \\ &\iff \text{left linear grammar for } L \end{aligned}$$

Consider the NFA constructed for 3.3.1 with the transitions inverted and the final and initial states swapped:

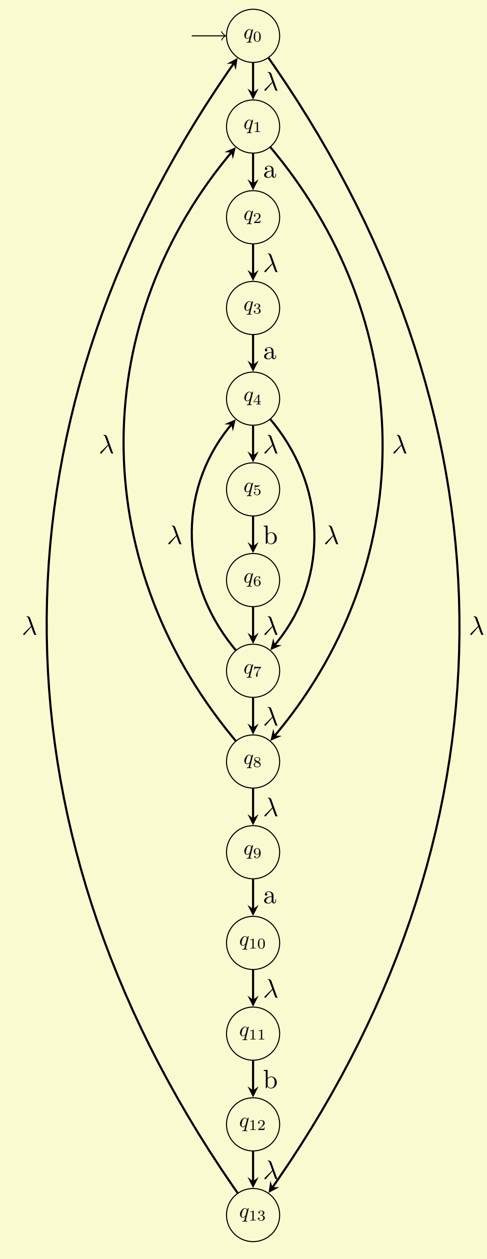


This NFA accepts the reverse of the original language. This NFA can be translated into a right-linear grammar. Now we reverse the left handside and the right handside. Note that reversing the left handside is irrelevant since reversing a single letter change nothing.

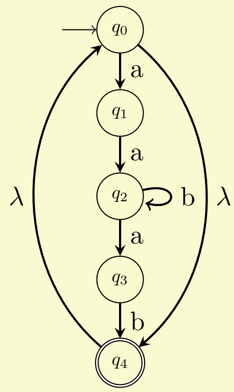
$$\begin{array}{ll} \Omega \to Zb & \Omega \to bZ \\ Z \to Bb & Z \to bB \\ B \to Ya & B \to aY \\ Y \to Ab & Y \to bA \\ A \to Xb \mid Ba & A \to bX \mid aB \\ X \to Sa & X \to aS \\ S \to \lambda & S \to \lambda \end{array}$$

Solution n°11

First we create a NFA corresponding to $(aab^*ab)^*$:



We can reduce this NFA and we construct the right-linear grammar:



$$
\begin{aligned}
&Q_0 \to aQ_1 | Q_5 \\
&Q_1 \to aQ_2 \\
&Q_2 \to bQ_2 \mid aQ_3 \\
&Q_3 \to bQ_4 \\
&Q_4 \to \lambda \mid Q_0
\end{aligned}
$$

Solution n°12

Find a regular grammar that generates the language

$$L = \{w \in \{a,b\}^* : n_a(w) + 3n_b(w) \text{ is even}\}$$

Define $E = 2\mathbb{N}$ the set of even and positive integers and $O = \mathbb{N} \setminus E$ the set of odd and positive integers.

- $n_a(w) \in E \Longleftrightarrow 3n_b(w) \in E \Longleftrightarrow n_b(w) \in E$
- $n_a(w) \in O \Longleftrightarrow 3n_b(w) \in O \Longleftrightarrow n_b(w) \in O$

$$\forall (n_a(w), n_b(w)) \in \mathbb{N} \times \mathbb{N} \ \ (n_a(w) + n_b(w)) \in E$$

This means that each word w in L is of even length.

$$S \to aaS \mid bb \mid ab \mid ba \mid \lambda$$