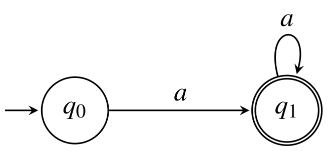
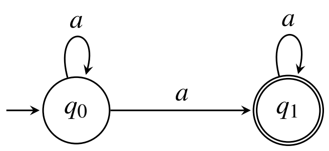
TD 5: Determinism

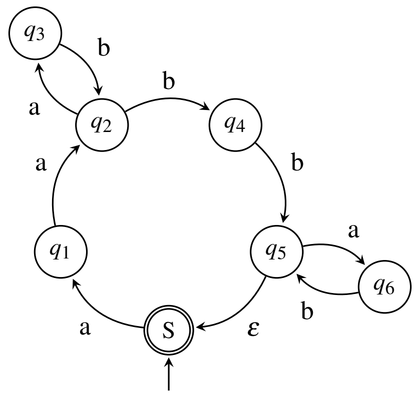
Exercise 1: Find two automata of minimal size recognizing the language $\{a^n \mid n > 0\}$ (note that one of them is not deterministic).
Solution:



and



Exercise 2: Let M be the following automaton (with one ε-transition) on the alphabet $\Sigma = \{a, b\}$.



1. Find a regular expression e such that $[\![e]\!] = [\![M]\!]$.
2. Draw a deterministic finite automaton M' (without ε-transition) such that $[\![M]\!] = [\![M']\!]$.
3. Draw a nondeterministic finite automaton M'' (without ε-transition) such that $[\![M']\!] = [\![M'']\!]$ and which is strictly smaller than M'.
4. Find a «natural» bijection between $[\![M]\!]$ and the finite lists of (ordered) pairs of natural numbers, i.e. the set $(\mathbb{N} \times \mathbb{N})^*$.

The regular expressions e_n (on the alphabet Σ) are inductively defined as follows:

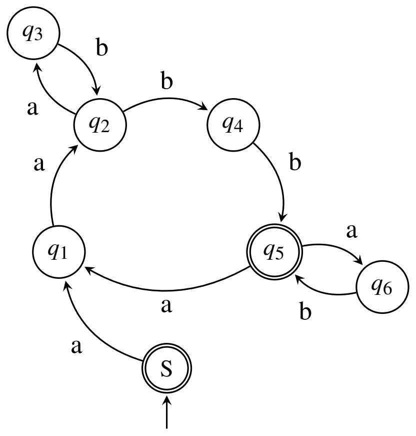
$$e_1 \quad = \quad (\mathrm{ab})^* \quad \text{and} \quad e_{n+1} = (\mathrm{a}^{2^n} e_n \mathrm{b}^{2^n} e_n)^*$$

5. For all $n \geqslant 1$, describe an automaton M_n (with ε-transitions) such that $[\![M_n]\!] = [\![e_n]\!]$.

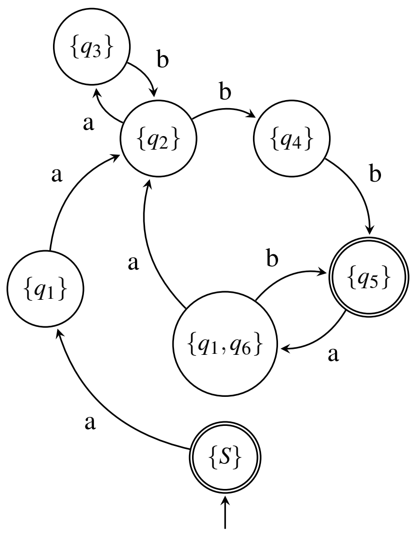
Solution:

1. $e = (aa(ab)^*bb(ab)^*)^*$

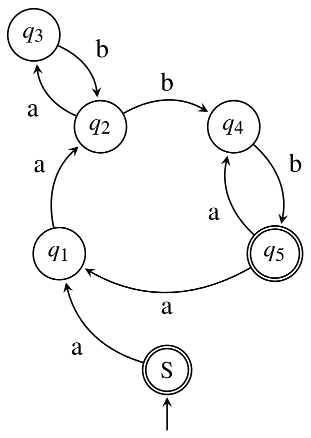
2. We start by removing the ε-transition in M. Do not forget that, since the ε-transition points to a final state, when we remove it, its source state q_5 has to become final too. We obtain the following automaton $\widehat{M}$.



The automaton $\widehat{M}$ above is non-deterministic: in state q_5, there are two outgoing transitions with label a. So, if we are in state q_5 and we read an 'a', we can go either to state q_1 or q_6.
We use the powerset construction to obtain the following deterministic automaton M'. In M', if we are in state $\{q_5\}$ and we read an 'a', we go to state $\{q_1, q_6\}$: this reflects what happens in $\widehat{M}$.



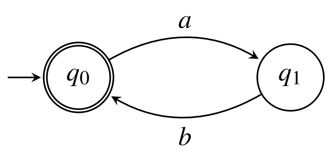
3. Notice that the automata M, $\widehat{M}$ and M' all have 7 states. Notice how in $\widehat{M}$, the states q_4 and q_6 play the same role: "when I see a b, go to state q_5". So we can remove one of them. Here is a non-deterministic automaton M'' recognizing the same language with only 6 states.



4. According to question 1, $[\![M]\!] = [\![(aa(ab)^*bb(ab)^*)^*]\!] = \{w_1 \cdots w_k \mid \forall i.\ w_i \in [\![aa(ab)^*bb(ab)^*]\!]\}$. So, a word $w \in [\![M]\!]$ is a finite sequence of words $(w_i)_{1 \le i \le k}$ such that each w_i is of the form $w_i = aa(ab)^n bb(ab)^m$, with $n, m \in \mathbb{N}$.
   To each of these w_i, we can bijectively associate the pair $f(w_i) = (n,m) \in \mathbb{N} \times \mathbb{N}$.
   And to $w = w_1 \cdots w_k$, we associate finite list $\phi(w) = f(w_1), \ldots, f(w_k) \in (\mathbb{N} \times \mathbb{N})^k$.

5. We define the automaton M_n by induction on n.

   - M_1 is the following automaton:

   - Assume M_n has been defined (and recognizes the language $[\![e_n]\!]$). We can build an automaton recognizing the language $\{a^{2^n}\}$ (this is a language with only one word, the automaton has $2^n + 1$ states), and one for the language $\{b^{2^n}\}$. We can use the procedure from lesson 3 (concatenation of two regular languages) to obtain an automaton for $[\![a^{2^n} e_n b^{2^n} e_n]\!]$, then we add an ε-transition from the final state to the initial state to obtain the Kleene star of this language. This gives us the automaton M_{n+1} recognizing $[\![e_{n+1}]\!]$.
     Note that, in particular, M_2 is the automaton M from the previous questions.

Exercise 3: Let F be a finite language on an alphabet Σ. Denote by L_F the language of words on Σ in which no word of F appears (i.e. for all $w \in F$ and all $\alpha, \beta \in \Sigma^*$, $\alpha w \beta \notin L_F$).

1. Find a *pattern* that matches L_F (i.e. a pattern e such that $[\![e]\!] = L$).

Assume that $\Sigma = \{a, b, c\}$ and $F = \{aa, bb, cc\}$.

2. Draw a deterministic finite automaton that recognizes L_F.

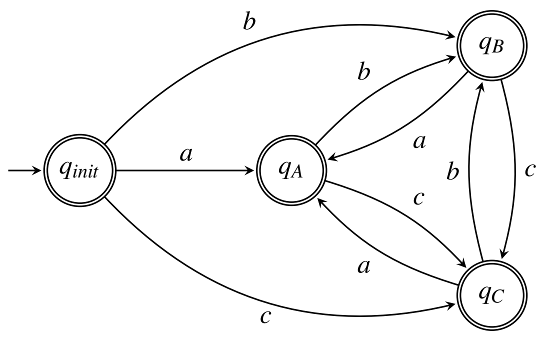
3. Find a *regular expression* that matches L_F. (Hint: use local definitions, e.g. x = (ab)*)

Solution:

1. Let $w_1, \ldots, w_k$ be the words of F. We choose the following pattern: $e = \sim (@(w_1 + \ldots + w_k)@)$.

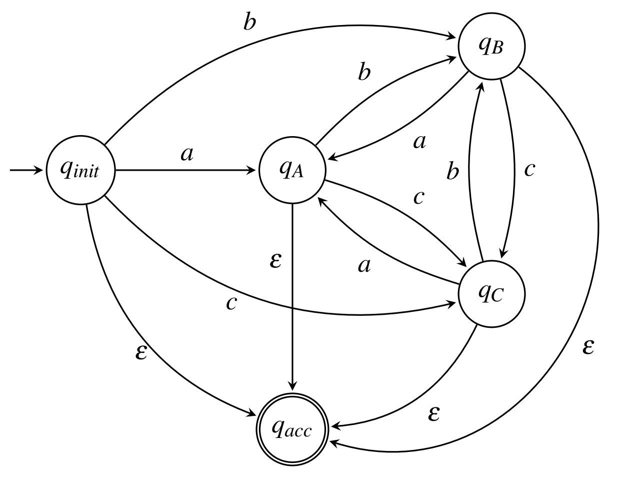
   Let us check that it works. By definition, since $[\![w_1 + \ldots + w_k]\!] = \{w_1, \ldots, w_k\} = F$, we have $[\![e]\!] = \Sigma^* \setminus (\Sigma^* \cdot F \cdot \Sigma^*)$. So, $[\![e]\!]$ is precisely the set of words which are not of the form $\alpha w \beta$ with $\alpha, \beta \in \Sigma^*$ and $w \in F$, which is what we wanted. Thus, $[\![e]\!] = L_F$.

2. With $\Sigma = \{a,b,c\}$ and $F = \{aa, bb, cc\}$, the following automaton recognizes L_F.

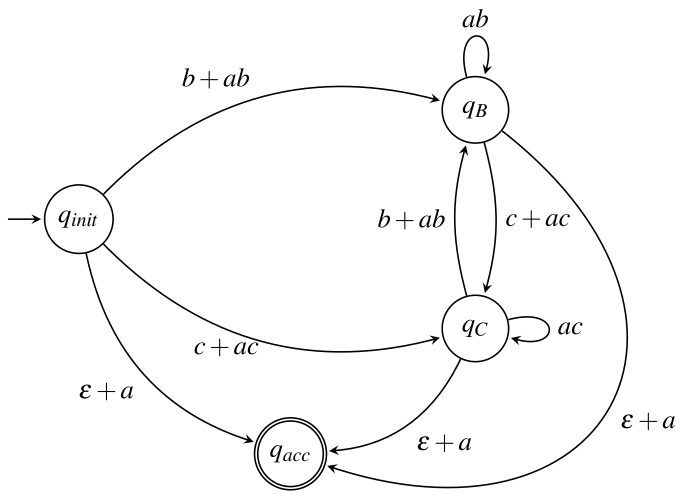


   It is straightforward to see that it works: intuitively, the meaning of state q_A is "I have just seen an a", in which case, only the letters b and c are allowed as next letters. Same for q_B and q_C.

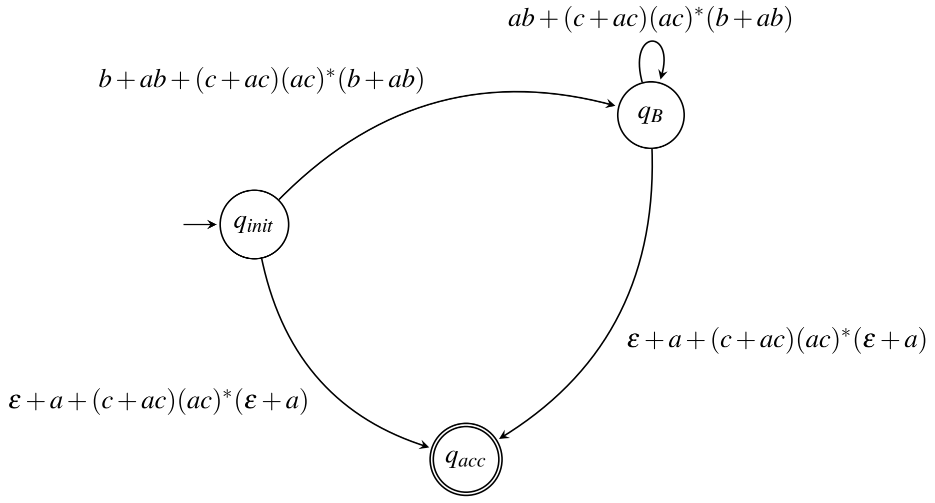
3. We want to use the Brzozowski-McCuskey state elimination procedure, but for this, we must first make sure that we have only one final state (because only the states that are not initial nor final can be eliminated). To do this, we add a new final state and ε-transitions as follows.



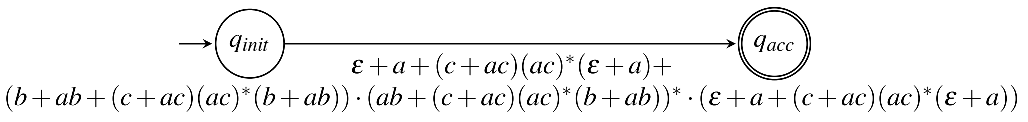
   First, we eliminate state q_A. It has 3 incoming transitions (coming from q_{init}, q_B, q_C) and 3 outgoing transitions (going to q_B, q_C, q_{acc}). Thus, when we delete q_A we are going to add $3 \times 3 = 9$ new edges. When this creates two parallel edges, we merge them by putting a '$+$' on the already existing edge.



Next, we eliminate state q_C. It has 2 ingoing transitions (from q_i and q_B) and 2 outgoing transitions (to q_B and q_{acc}) and a loop, so we should create 4 new transitions. As before, we add the corresponding regular expression to the already existing edge labels.


$ab+(c+ac)(ac)^*(b+ab)$
$b+ab+(c+ac)(ac)^*(b+ab)$
q_B
q_{init}
$\varepsilon+a+(c+ac)(ac)^*(\varepsilon+a)$
$\varepsilon+a+(c+ac)(ac)^*(\varepsilon+a)$
q_{acc}

Finally, we eliminate state q_B and get the following automaton:


q_{init}
q_{acc}
$\varepsilon+a+(c+ac)(ac)^*(\varepsilon+a)+$
$(b+ab+(c+ac)(ac)^*(b+ab))\cdot(ab+(c+ac)(ac)^*(b+ab))^*\cdot(\varepsilon+a+(c+ac)(ac)^*(\varepsilon+a))$

The regular expression on the edge matches the language L_F.

Remark: if we decided to eliminate the states in a different order, we would have obtained a different regular expression in the end; but both of them would be equivalent since they match the language L_F (by correctness of the state elimination procedure).

Exercise 4: Let Σ be the alphabet $\{a,b\}$. Here, the letters of 'a' and 'b' stand for the opening and the closing parentheses, to distinguish them from '(' and ')' which appears in regular expressions. Then, a word w on Σ is said to be well-parenthesized when it is recognized by the following grammar.

$$S \to SS \qquad S \to aSb \qquad S \to \varepsilon$$

The *depth* of such a word is inductively defined as follows:

$$\text{depth}(\varepsilon) = 0$$

and for all well-parenthesized words w and w'

$$\text{depth}(ww') = \max(\text{depth}(w), \text{depth}(w')) \quad \text{and} \quad \text{depth}(awb) = \text{depth}(w) + 1$$

1. Find a deterministic automaton on Σ which recognizes the language

$$L = \{w \in \Sigma^* \mid w \text{ is well-parenthesized and depth}(w) \leqslant 2\}$$

2. Find a regular expression that matches L.

3a. Find a mapping $h : \{0,1\} \to \Sigma^*$ that «induces» a bijection from the language of binary words (i.e. $\{0,1\}^*$) to the language

$$\{w \in \Sigma^* \mid awb \in L\}$$

3b. Deduce a regular expression that matches L and whose star height is 1.

3c. What is the star height of the language L?

4. Let e_1 be the regular expression $(\text{ab})^*$, and define inductively the regular expression e_{n+1} as $(ae_nb)^*$. Prove that for $n \geqslant 2$, the star height of $[\![e_n]\!]$ is at most $n-1$.

Solution: Remember the following characterization of well-parenthesized words. Given a word w, we use the following procedure. Start a counter at 0, then read the word w from left to right. When the next letter is an opening parenthesis (in this case, the symbol 'a'), we add 1 to the counter, and when it is a closing parenthesis (here, 'b'), we subtract 1 to the counter. Then w is well-parenthesized *iff* the counter always remains positive, and at the end of the word the counter is 0.

Formally we can state this as follows. Write $|w|_a$ for the number of a's in w, and similarly for the number of b. Then w is well-parenthesized *iff*

- $|w|_a = |w|_b$, and
- for every prefix x of w, $|x|_a \geq |x|_b$.

We have a similar characterization of well-parenthesized words of depth $\leq k$, except that the counter must remain between 0 and k.

Lemma: The word w is well-parenthesized of depth $\leq k$ *iff* $|w|_a = |w|_b$ and for every prefix x of w, $|x|_b \leq |x|_a \leq |x|_b + k$.

Proof. [1] $(\Rightarrow)$: Let w be a well-parenthesized word of depth $\leq k$. By definition, there is a derivation $S \to^* w$ in the grammar defined above. We are going to prove the result by induction on the number of steps of the derivation. We do a case disjunction depending on what the first rule applied is: either $S \to SS$, or $S \to aSb$, or $S \to \varepsilon$.

[1]The proof of this Lemma is not part of the exercise, but it a very good example of the kind of inductive reasoning we can do when we want to prove things about grammars; in particular the $(\Rightarrow)$ direction.

- $S \to \varepsilon$: then $w = \varepsilon$, and we indeed have $|\varepsilon|_a = 0 = |\varepsilon|_b$, and it is also true for the only prefix of ε (which is ε itself).
- $S \to SS \to^* w$: then $w = w_1 w_2$, where both w_1 and w_2 are well-parenthesized. By definition of the depth, $\text{depth}(w) = \max(\text{depth}(w_1), \text{depth}(w_2))$, and since w is of depth $\leq k$, then both w_1 and w_2 must also have a depth $\leq k$. Moreover, both w_1 and w_2 are obtained from shorter derivations, so we can use the induction hypothesis on them:
  - IH_{w_1}: $|w_1|_a = |w_1|_b$ and for every prefix x of w_1, $|x|_b \leq |x|_a \leq |x|_b + k$
  - IH_{w_2}: $|w_2|_a = |w_2|_b$ and for every prefix x of w_2, $|x|_b \leq |x|_a \leq |x|_b + k$

  Thus, we have the first property $|w|_a = |w_1|_a + |w_2|_a = |w_1|_b + |w_2|_b = |w|_b$.

  Moreover, let x be a prefix of w. Then, either x is a prefix of w_1, so by induction we have $|x|_b \leq |x|_a \leq |x|_b + k$ as required. Or x is of the form $w_1 x'$ where x' is a prefix of w_2, and then we have $|x|_b = |w_1|_b + |x'|_b \underbrace{\leq}_{\text{by } IH_{w_2}} |w_1|_b + |x'|_a \underbrace{=}_{\text{by } IH_{w_1}} |w_1|_a + |x'|_a = |x|_a$; and similarly we obtain $|x|_a \leq |x|_b + k$.
- $S \to aSb \to^* w$: then $w = aw'b$, where w is well-parenthesized. By definition, $\text{depth}(w) = \text{depth}(w') + 1$, so w' has depth $\leq k-1$. Since w' is obtained from a shorter derivation, so we have the following induction hypothesis:
  - $IH_{w'}$: $|w'|_a = |w'|_b$ and for every prefix x of w', $|x|_b \leq |x|_a \leq |x|_b + k - 1$

  We check the two properties for w: $|w|_a = 1 + |w'|_a = 1 + |w'|_b = |w|_b$.

  And given a prefix x of w, either $x = a$ (in which case we easily check that the property holds), or $x = ax'$ where x' is a prefix of w'. Then $|x|_b = |x'|_b \leq |x'|_a \leq |x|_a$, and $|x|_a = 1 + |x'|_a \leq 1 + |x'|_b + k - 1 = |x'|_b + k = |x|_b + k$.

($\Leftarrow$) : (Sketch). Assume w is such that $|w|_a = |w|_b$ and for every prefix x of w, $|x|_b \leq |x|_a \leq |x|_b + k$. We prove by induction on the length of w that w is well-parenthesized of depth $\leq k$.

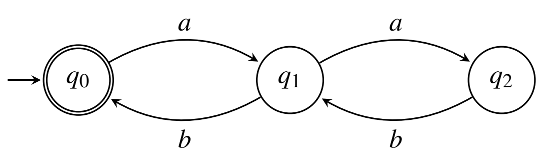
- $w = \varepsilon$: OK.
- Let x be the smallest prefix of w such that $|x|_a = |x|_b$. Then the first letter of x must be an 'a' and the last letter of x must be a 'b'. Thus, $x = ax'b$ and the word w can be decomposed as $w = ax'by$.

  Then, we can check that:
  - $|x'|_a = |x'|_b$ and for every prefix z of x', $|z|_b \leq |z|_a \leq |z|_b + k - 1$
  - $|y|_a = |y|_b$ and for every prefix z of y, $|z|_b \leq |z|_a \leq |z|_b + k$

  By induction hypothesis on x' and y (which are smaller than w), we obtain that x' is well-parenthesized of depth $\leq k-1$, and y is well-parenthesized of depth $\leq k$. Then, w is also well-parenthesized (with a derivation of the form $S \to SS \to aSbS \to^* ax'by$), and $\text{depth}(w) = \max(\text{depth}(x') + 1, \text{depth}(y)) \leq k$.

□

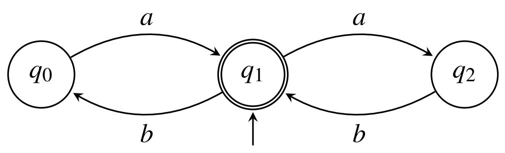
1. We use the Lemma above to write the automaton for the well-parenthesized words of depth ≤ 2. The states q_0, q_1, q_2 correspond to the value $0, 1, 2$ of the counter. According to the Lemma, the counter can never go above 2.



2. Using the state elimination procedure (first eliminate q_2 then q_1), we obtain that L is recognized by the regular expression $e = (a(ab)^*b)^*$.

3a. Let $L' = \{w \in \Sigma^* \mid awb \in L\}$. Note that the elements of L' might not be well-parenthesized: for example, since $abab \in L$, we have $ba \in L'$.

Let us see what the "counter procedure" does on a word of the form $awb \in L$. First, we start with counter 0 and after the first a, the counter is 1. So, we star the word w with a counter equal to 1. At the end of the word, the counter must be 0, so, before the last b, the counter was 1. Thus, at the end of the word w, the counter must be 1 too.
Therefore, L' is recognized by the following automaton:



Using the state elimination procedure, L' is recognized by the regular expression $(ab)^* + (ba)^*$. So, a word $w \in L'$ can be decomposed as a sequence $w = w_1 \dots w_k$ where each w_i is either ab or ba. Replacing every occurrence of ab by 0 and every occurrence of ba by 1, we obtain a bijection between $\{0,1\}^*$ and L'.

Remark: This bijection is a special kind of map: it is obtained from the mapping $h : \{0,1\} \to \Sigma^*$ defined as $h(0) = ab$ and $h(1) = ba$. The mapping h extends in a unique way to a language morphism $h^* : \{0,1\}^* \to \Sigma^*$ (see tutorial 2, exercise 8, question 3): given a sequence $b = b_1 \dots b_k \in \{0,1\}^*$, where each b_i is 0 or 1, its image is $h^*(b) = h(b_1) \cdots h(b_k)$.

3b. The expression $a(ab+ba)^*b$ matches L

3c. The star-height of L is ≤ 1 since we have found a regular expression of star-height 1 which recognizes L. It cannot be 0 since the language is infinite (see tutorial 4, exercise 3, question 1). So, the star-height is 1.

4. By induction on n:

- $n = 2$: Note that e_2 is the expression that we found in question 2. According to question 3c, the star-height of $[\![e_2]\!]$ is 1.
- Assume by induction hypothesis that the star-height of $[\![e_n]\!]$ is at most $n-1$. So, there exists a regular expression e'_n of star-height $\leq n-1$ such that $[\![e'_n]\!] = [\![e_n]\!]$.
  Now consider the regular expression $e'_{n+1} = (ae'_n b)^*$. The star-height of this expression is $1 + \mathsf{sh}(e'_n) \leq n$. Moreover, $[\![e'_{n+1}]\!] = (\{a\} \cdot [\![e'_n]\!] \cdot \{b\})^* = (\{a\} \cdot [\![e_n]\!] \cdot \{b\})^* = [\![e_{n+1}]\!]$. Thus, the language $[\![e_{n+1}]\!]$ has star-height $\leq n$.