Exercise sessions #3 and #4
Regular languages

prof. B. Demoen W. Van Onsem

March-April 2015

Exercise 1. For a given string $w = w_1w_2\ldots w_n$ we denote the reversed string as $w^{\mathcal{R}} = w_n \ldots w_2w_1$. For a given language L we write $L^{\mathcal{R}} = \left\{w^{\mathcal{R}} \mid w \in L\right\}$. Given L is regular, is $L^{\mathcal{R}}$ regular as well?

Answer. Yes. we can prove this using two different strategies: (1) providing a construction that for every regular expression e constructs a regular expression $e^{\mathcal{R}}$ such that the language described by $e^{\mathcal{R}}$ is the reverse of the the original language; (2) provide a construction that generates for a given DFA D an NFA $N^{\mathcal{R}}$ that decides the reverse language.

1. Regular expression that describes the reverse language: we can do this inductively on the structure of a regular expression:

$$\begin{aligned}
\epsilon^{\mathcal{R}} &= \epsilon && (1)\\
\phi^{\mathcal{R}} &= \phi && (2)\\
\forall a \in \Sigma : a^{\mathcal{R}} &= a && (3)\\
\left(E_1E_2\right)^{\mathcal{R}} &= \left(E_2^{\mathcal{R}}E_1^{\mathcal{R}}\right) && (4)\\
\left(E_1|E_2\right)^{\mathcal{R}} &= \left(E_1^{\mathcal{R}}|E_2^{\mathcal{R}}\right) && (5)\\
\left(E_1^{\star}\right)^{\mathcal{R}} &= \left(E_1^{\mathcal{R}}\right)^{\star} && (6)
\end{aligned}$$

2. Automaton that decides the reverse language:

   Construction 1. Given a DFA $D = \langle Q, \Sigma, \delta, q_s, F\rangle$. We consider an addition state $q' \notin Q$ that does not belong to the original set of states Q. We consider a new transition function δ' such that:

$$\begin{aligned}
\forall q_i \in Q, \sigma \in \Sigma : \delta'\left(q_i, \sigma\right) &= \left\{q_j \in Q \mid q_i = \delta\left(q_j, \sigma\right)\right\} && (7)\\
\delta'\left(q', \epsilon\right) &= F && (8)
\end{aligned}$$

   De constructed NFA is defined as: $N^{\mathcal{R}} = \langle Q \cup \{q'\}, \Sigma, \delta', \{q'\}, q_s\rangle$

◇

Exercise 2. Give for every language below over the alphabet $\Sigma = \{0, 1\}$ a regular expression that determines the language. Construct an NFA that decides the language.

1. $\{w \mid \text{on every odd position of } w \text{ we write a } 1\}$
2. $\{w \mid w \text{ contains at least two 0s and at most one } 1\}$
3. The complement language of $\{11, 111\}$
4. $\{w \mid w \text{ contains an equal amount the substring 01 and the substring 10}\}$

Answer.

1. $\left(1\left(0|1\right)\right)^{\star}\left(1|\epsilon\right)$
2. $0^{\star}\left(00|001|010|100\right)0^{\star}$

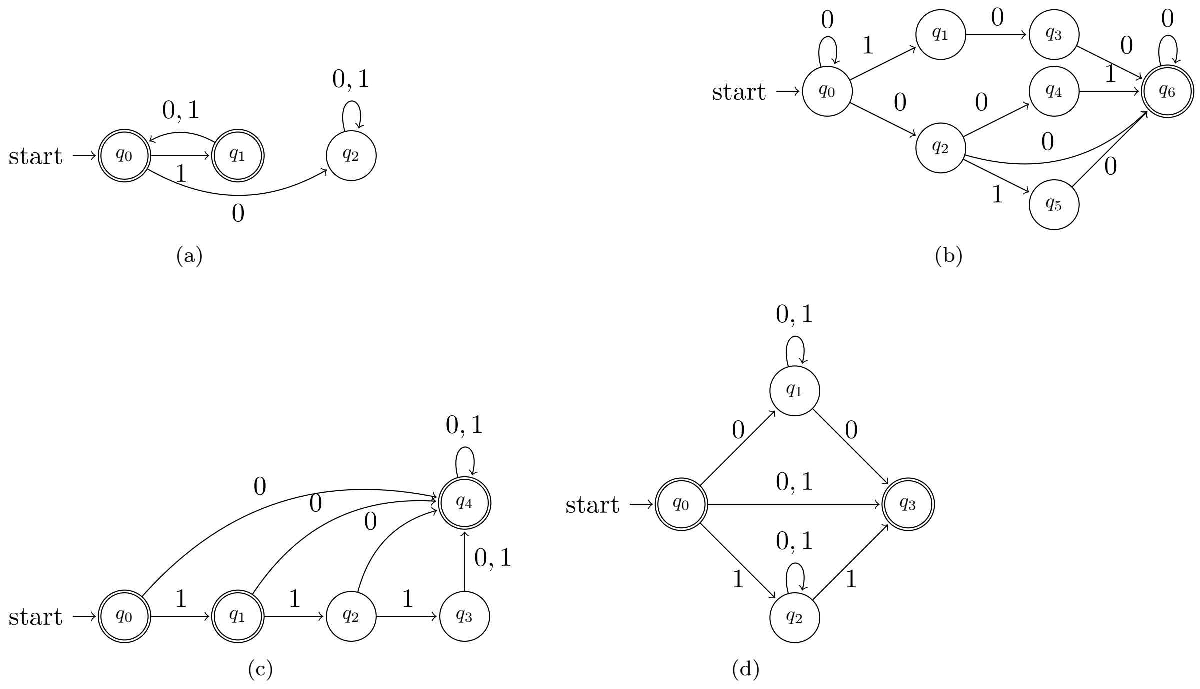


Figure 1: NFAs for the given languages.

.

3. $\epsilon|0\,(0|1)^\star\,|10\,(0|1)^\star\,|110\,(0|1)^\star\,|111\,(0|1)\,(0|1)^\star$
4. $0^\star|1^\star|0\,(0|1)^\star\,0|1\,(0|1)^\star\,1$

For NFAs: see Figure 1. $\diamond$

Exercise 3. Prove that for every $n \geq 1$ there exists at least one NFA that decides the following languages.

1. $\left\{a^k \mid k \text{ is a multiple of } n\right\}$ over the alphabet $\Sigma = \{a\}$.
2. $\{x \mid x \text{ is the binary representation of a natural number that is a multiple of } n\}$

Answer.

1. One can construct a DFA that decides a language L_n for a given n as follows:

   Construction 2. One considers n states: $q_1, q_2, \ldots, q_n$. q_n is the initial state and the only accepting state ($F = \{q_n\}$). The transition function δ is defined as follows:

   $$\forall q_i \in Q, j \in \{0,1\} : \delta_1\left(q_i, a\right) = \begin{cases} q_{i+1} & i < n \\ q_1 \end{cases} \tag{9}$$

   The NFA is defined as $\langle \{q_1, q_2, \ldots, q_n\}, \{a\}, \delta_1, q_n, \{q_n\} \rangle$.

2. We assume the binary representation uses *little endian*: the least significant bits occur at the end of the string. The endianness is of no vital importance: the first exercise already shows that if a language is regular, it's reverse is regular as well, we can thus construct the reverse language for the *big endian* variant. One can construct a DFA that decides the language as follows:

   Construction 3. One considers n states: $q_0, q_1, \ldots, q_{n-1}$. q_0 is the initial state and the only accepting state. The transition function δ is defined as follows:

   $$\forall q_i \in Q, j \in \{0,1\} : \delta_2\left(q_i, j\right) = q_k k = 2 \cdot i + j \mod n \tag{10}$$

   The NFA is defined as $\langle \{q_0, q_1, \ldots, q_{n-1}\}, \{0,1\}, \delta_2, q_0, \{q_0\} \rangle$.

⋄

Exercise 4. Prove that given a regular language L over an alphabet Σ, the language

$$L^c = \{w \mid w \text{ is a string over } \Sigma \text{ and } w \notin L\} \tag{11}$$

is regular as well.

Answer. Since L is a regular language, there exists a DFA $D = \langle Q, \Sigma, \delta, q_s, F\rangle$ that decides the language. Consider the DFA $D' = \langle Q, \Sigma, \delta, q_s, Q \setminus F\rangle$. For every string s over Σ there is exactly one sequence of states $r_1 r_2 r_3 \ldots r_{n+1}$ such that $r_1 = q_s$ and $\delta(r_i, s_i) = r_{i+1}$. This sequence remains the same for both D and D' (since they share the same initial state and transition function). A DFA accepts a string if and only if $r_{n+1} \in F$. Since the set of accepting states for D' is $F' = Q \setminus F$ this implies that if D accepts, D' will reject the string and vice versa. ⋄

Exercise 5. Given the alphabet Σ_3:

$$\Sigma_3 = \left\{\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \ldots, \begin{bmatrix}1\\1\\1\end{bmatrix}\right\} \tag{12}$$

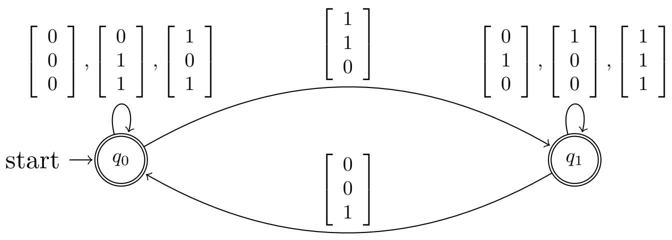
Consider a string over Σ_3 as three rows of 0's and 1's and consider every row to be the binary representation of a binary number (*little-endian*). We define L as the language

$$\{w \mid \text{the last row of } w \text{ is the sum of the two rows above.}\} \tag{13}$$

over Σ_3. Thus $\begin{bmatrix}0\\0\\1\end{bmatrix}\begin{bmatrix}1\\0\\0\end{bmatrix}\begin{bmatrix}1\\1\\0\end{bmatrix}$ is an element of L, but $\begin{bmatrix}0\\0\\1\end{bmatrix}\begin{bmatrix}1\\0\\1\end{bmatrix}$ isn't. Show that L is a regular language.

Answer. First we make two assumptions, later in this answer we show how we can modify the constructed NFA such that we we can construct NFAs for different flavors of this exercise:

1. We read the string right-to-left (thus *big-endian*) with the least significant bits first; en
2. The NFA works with *wraparound*: when the number of bits is not sufficient to represent the sum, we drop the most significant bits.



$\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix}$
$\begin{bmatrix}1\\1\\0\end{bmatrix}$
$\begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\1\end{bmatrix}$
start
q_0
q_1
$\begin{bmatrix}0\\0\\1\end{bmatrix}$

Figure 2: An NFA that decides binary addition.

The NFA has two states: q_0 and q_1. The states represents the current *carry*. We work from right-to-left, so for the first addition the carry is 0. When we process $\begin{bmatrix}0\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\1\\1\end{bmatrix}$ or $\begin{bmatrix}1\\0\\1\end{bmatrix}$ these are valid additions that do not "generate" carry: the system remains in state q_0. For $\begin{bmatrix}1\\1\\0\end{bmatrix}$ we will generate carry, we thus make the transition to q_1. In state q_1 there are two possibilities: characters that "propagate" (or even generate) carry: $\begin{bmatrix}0\\1\\0\end{bmatrix}$, $\begin{bmatrix}1\\0\\0\end{bmatrix}$, $\begin{bmatrix}1\\1\\1\end{bmatrix}$. The other possibility is that the carry is lost due to the fact that the first rows don't contain a 1, this happens for $\begin{bmatrix}0\\0\\1\end{bmatrix}$. The NFA doesn't accept strings if the NFA is in a state and no edge contains the character it is supposed to process.

In the first exercise, we've shown that we can reverse any regular language, we can use this concept to reverse the *big endian* format to *little endian*. We can disable wraparound by making q_1 a non-accepting state. $\diamond$

Exercise 6. Prove that given L_1 and L_2 are regular languages, $L_1 \cap L_2$ is a regular language as well. Is $L_1 \setminus L_2$ a regular language?

Answer. We can prove this by first defining two constructions:

1. Intersection $(L_1 \cap L_2)$

   Construction 4. Given two DFAs $\langle Q_1, \Sigma, \delta_1, q_{s,1}, F_1\rangle$ and $\langle Q_2, \Sigma, \delta_2, q_{s,2}, F_2\rangle$, we consider a DFA $\langle Q, \Sigma, \delta, q_s, F\rangle$ that decides the intersection of the two languages:

$$\begin{aligned} Q &= Q_1 \times Q_2 = \{\langle q_1, q_2\rangle \mid q_1 \in Q_1 \wedge q_2 \in Q_2\} && (14)\\ F &= F_1 \times F_2 && (15)\\ q_s &= \langle q_{s,1}, q_{s_2}\rangle && (16)\\ \forall q_1 \in Q_2, q_2 \in Q_2, \sigma \in \Sigma : \delta\left(\langle q_1, q_2\rangle, \sigma\right) &= \left\langle \delta_1\left(q_1, \sigma\right), \delta_2\left(q_2, \sigma\right)\right\rangle && (17) \end{aligned}$$

   The set of states Q contains thus every possible couple of two original states (from different DFA's). For every character, we "update" both states according to the corresponding DFA transition functions δ_1, δ_2. We only accept if the couple of the final state consist of accepting states. Since a DFA is deterministic, there exists only one path for every string. Given a string s where the sequence of states for the first DFA would be $r_{1,0}r_{1,1}r_{1,2}\ldots r_{1,m+1}$ and the second $r_{2,0}r_{2,1}r_{2,2}\ldots r_{2,m+1}$, the path for the constructed DFA is thus: $\langle r_{1,0}, r_{2,0}\rangle \langle r_{1,1}, r_{2,1}\rangle \ldots \langle r_{1,m+1}, r_{2,m+1}\rangle$. Since the first and second DFA only accept of $r_{1,m+1}$ and $r_{2,m+1}$ are accepting states, the constructed DFA will only accept if the two DFA's both accept.

2. Set difference $(L_1 \setminus L_2)$

   Construction 5. Given two DFAs $\langle Q_1, \Sigma, \delta_1, q_{s,1}, F_1\rangle$ and $\langle Q_2, \Sigma, \delta_2, q_{s,2}, F_2\rangle$, we consider a DFA $\langle Q, \Sigma, \delta, q_s, F\rangle$ that decides the set difference:

$$\begin{aligned} Q &= Q_1 \times Q_2 = \{\langle q_1, q_2\rangle \mid q_1 \in Q_1 \wedge q_2 \in Q_2\} && (18)\\ F &= F_1 \times (Q \setminus F_2) && (19)\\ q_s &= \langle q_{s,1}, q_{s_2}\rangle && (20)\\ \forall q_1 \in Q_2, q_2 \in Q_2, \sigma \in \Sigma : \delta\left(\langle q_1, q_2\rangle, \sigma\right) &= \left\langle \delta_1\left(q_1, \sigma\right), \delta_2\left(q_2, \sigma\right)\right\rangle && (21) \end{aligned}$$

   The construction is almost unique to the first one. We only modify the set of accepting states: now it contains all couples of accepting states for the first and non-accepting states for the second. Based on the exercise on the complement language, one can prove correctness. $\diamond$

Exercise 7. An *all-paths*-NFA $\langle Q, \Sigma, \delta, q_0, F\rangle$ differs from an NFA because it accepts a string w only if

- for every possible subdivision $w = y_1y_2\ldots y_m$, $y_i \in \Sigma_\varepsilon$ and every state sequence $r_0, r_1, \ldots, r_m$ such that $r_0 = q_0$ and $r_{i+1} \in \delta(r_i, y_{i+1})$, it holds that $r_m \in F$;
- there is at least one subdivision $w = y_1y_2\ldots y_m$, $y_i \in \Sigma_\varepsilon$ such that there is a state sequence $r_0, \ldots, r_m$ with $r_0 = q_0$ and $r_{i+1} \in \delta(r_i, y_{i+1})$.

Show that L is regular, if and only if there exists an all-paths-NFA that decides L.

Answer. We prove this in both directions:

1. Given L is a regular language, the language is recognized by an all-path NFA.

   Proof. Given L is a regular language, then there exists a DFA $D = \langle Q, \Sigma, \delta, q_s, F\rangle$ that decides L. A DFA is an NFA with a determinstic transition function. This implies that for a given string s, there is exactly once path of states. We can transform D into an equivalent NFA $N = \langle Q, \Sigma, \delta', q_s, F\rangle$:

$$\forall q \in Q, \sigma \in \Sigma : \delta'\left(q, \sigma\right) = \left\{\delta\left(q, \sigma\right)\right\} \qquad (22)$$

Given N is always in one state at the time, and equivalent to D, for every string s there exists one path of states as well. When the path of states for s is leading to an accepting state, D, The all-path NFA N will accept as well since there is one such path of states and it is accepting: all paths thus accepted and vice versa. □

2. Given L is recognized by an all-paths-NFA, L is regular.

   Proof. We prove this by using a construction that converts an all-path-NFA into a DFA that recognizes the same language.

   Construction 6. Given an all-paths-NFA $\langle Q, \Sigma, \delta, q_0, F\rangle$ we construct a DFA $\langle \mathcal{Q}, \Sigma, \delta', q_0', \mathcal{F}\rangle$ that decides the same language:

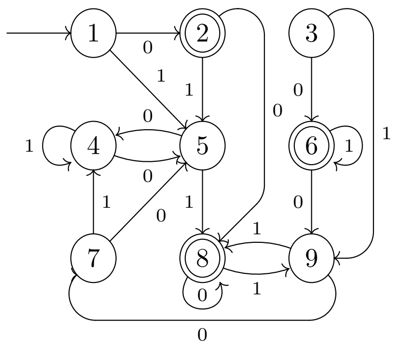
$$\mathcal{Q} = \mathcal{P}(Q) \tag{23}$$
$$\mathcal{F} = \mathcal{P}(F) \setminus \{\emptyset\} \tag{24}$$
$$q_0' = \text{eb}(q_0) \tag{25}$$
$$\forall Q' \in \mathcal{Q}, \sigma \in \Sigma : \delta'(Q', \sigma) = \delta_d(Q', \sigma) \tag{26}$$

   With eb and δ_d defined as in the course text.

   We approximately perform the same transition as from a normal NFA to a DFA. In every state of the DFA, the “heads” of the paths are stored. We modify $\mathcal{F}$ such that a state in the DFA is accepting if and only if all stored heads are accepting, and there is at least one such head. □

◇

Exercise 8. Minimize the following DFA



Answer. One minimizes a DFA in two phases:

1. Remove unreachable states; and
2. Combine f-equivalent states.

The two phases can be performed in arbitrary order, but the given order will in many cases save us some work.

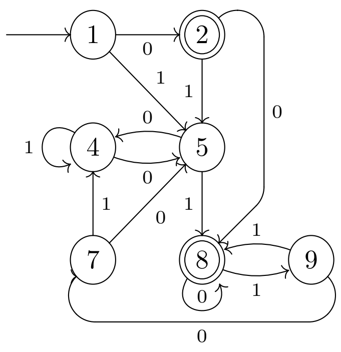
Unreachable states We calculate which states are reachable from the initial state. We do this by performing *fixed-point arithmetic* on the following function:

$$\text{reaches} : \mathcal{P}(Q) \to \mathcal{P}(Q) : R \mapsto R \cup \{\delta(q, \sigma) \mid q \in R, \sigma \in \Sigma\}. \tag{27}$$

The fixed-point process starts with the singleton containing the initial state of the automaton, consecutively we calculate:

$$R_0 = \{1\} \tag{28}$$
$$R_1 = \{1, 2, 5\} \tag{29}$$
$$R_2 = \{1, 2, 4, 5, 8\} \tag{30}$$
$$R_3 = \{1, 2, 4, 5, 8, 9\} \tag{31}$$
$$R_4 = \{1, 2, 4, 5, 7, 8, 9\} \tag{32}$$
$$R_5 = R^\star = \{1, 2, 4, 5, 7, 8, 9\} \tag{33}$$

The unreachable states are thus $Q \setminus R^\star = \{3, 6\}$. We remove these states as well as the edges originating from these states. The edges that end in the unreachable states can only originate from unreachable states as well. The DFA after reachability analysis is thus:



f-equivalent states First we construct a graph with f-different states. Because in general such graph has a large number of edges, we will denote the graph using a table. First we consider every two states such that exactly one of the states is an element of F. The table looks like:

	1	2	4	5	7	8	9
1		ϵ				ϵ	
2	ϵ		ϵ	ϵ	ϵ		ϵ
4		ϵ				ϵ	
5		ϵ				ϵ	
7		ϵ				ϵ	
8	ϵ		ϵ	ϵ	ϵ		ϵ
9		ϵ				ϵ	

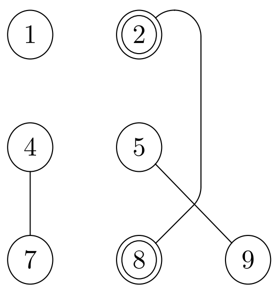
We now iterate over every pair of states (such that the corresponding element in the table is empty), and add, in case there is a conflict the symbol to the table:

	1	2	4	5	7	8	9
1		ϵ	0	01	0	ϵ	01
2	ϵ		ϵ	ϵ	ϵ		ϵ
4	0	ϵ		1		ϵ	1
5	01	ϵ	1		1	ϵ	
7		ϵ		1		ϵ	1
8	ϵ		ϵ	ϵ	ϵ		ϵ
9	01	ϵ	1		1	ϵ	

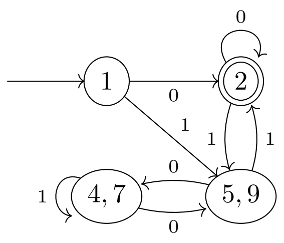
Strictly speaking it is not necessary to fill in the character: it suffices to know there is a conflict. Furthermore the table is symmetric: element i, j is always equivalent with element j, i, this saves approximately half of the work. Finally the diagonal will always be empty: a state can never have a conflict with itself. Since we made modifications in the previous iteration of the process, we need to run a second iteration:

	1	2	4	5	7	8	9
1		ϵ	01	01	01	ϵ	01
2	ϵ		ϵ	ϵ	ϵ		ϵ
4	01	ϵ		01		ϵ	01
5	01	ϵ	01		01	ϵ	
7	01	ϵ		01		ϵ	01
8	ϵ		ϵ	ϵ	ϵ		ϵ
9	01	ϵ	01		01	ϵ	

The fact that symbol are added to an element that contained already one or more symbols, doesn't matter. Since there are no elements that contained no symbols in the previous iteration, but now do, we no longer need to search for conflicts. Now we remove the edges between the states that don't have a conflict: this is the case for the following tuples: $\{\langle 4,7\rangle, \langle 5,9\rangle, \langle 2,8\rangle\}$. The complement graph is thus:



There is no clique of 3 or more vertices, the cliques (Q_i's) are thus $\{\langle 4,7\rangle, \langle 5,9\rangle, \langle 2,8\rangle\}$. We simple merge these states in the original DFA and modify the edges: if an edge pointed to one of the elements in the clique, it now points to the clique-state and edges originating from the states in a clique evidently point to the same state or clique. The minimal DFA is thus:



◇

Exercise 9. Consider the following alphabets $\Sigma_1 = \{0,1\}, \Sigma_2 = \{a\}$ en $\Sigma_3 = \{a,b,c\}$. Show - using the pumping lemma - that the following languages are not regular:

1. $\{w \in \Sigma_1^* \mid |w| \text{ is even and the 1's only occur in the second half of } w\}$
2. $\{a^{(2^n)} \in \Sigma_2^* \mid n \geq 0\}$
3. $\{a^n \in \Sigma_2^* \mid n \text{ is a prime number}\}$
4. $\{0^m 1^n \in \Sigma_1^* \mid m \neq n\}$
5. $\{w \in \Sigma_1^* \mid w \text{ isn't a palindrome}\}$
6. $\{a^i b^j c^k \in \Sigma_3^* \mid i,j,k \geq 0 \text{ such that given } i=1\text{, holds, } j=k \text{ must hold as well}\}$

Answer.

1. We consider $L' = L^{\mathcal{R}}$: the language

$$L' = \{w \in \Sigma_1^* \mid |w| \text{ is even en de 1'en in } w \text{ komen enkel voor in de eerste helft van } w\} \quad (34)$$

Proof. We now consider the string $s = 1^p 0^p$ with p the unknown pumping length. It is clear that $s \in L'$. For every valid breakdown $s = xyz$ that satisfies the conditions of the pumping lemma, we know:

(a) $x \in 1^\star$

(b) $y \in 11^\star$ met $|y| = l > 1$

When we "pump" the string, we thus generate $xy^2z = 1^{p+l}0^p$. Since $l > 0$ we generate a string with 1's outside the first half of the string. Therefor $xy^2z \notin L$, regardless of the way we breakdown s in x, y and z. □

Because the reverse language of a regular language is a regular language, it means that the reverse of L' (the original language) is not regular as well. Thus L is not regular.

2. $L = \{a^{(2^n)} \in \Sigma_2^* \mid n \geq 0\}$

   Proof. Consider the string a^{2^p} with p the unknown pumping length. For every valid breakdown $s = xyz$ it holds that $y \in aa^\star$ en $l = |a|$ with $0 < l \leq p$. Therefore $xy^2z = a^{2^p+l}$. $2^p + l$ is only a power of 2 given $l = 2^p$ or greater, but because $l \leq p$ and $\forall i \in \mathbb{N} : i < 2^i$ we know that $l < 2^p$, thus $2^p + l$ is not a power of 2 and $xy^2z \notin L$, regardless of the way we break s down into substrings. Thus L is not regular. □

3. $L = \{a^n \in \Sigma_2^* \mid n \text{ is a prime number}\}$

   Proof. Consider the string a^q with $q > p$, p the unknown pumping length and q a prime number. For every valid breakdown $s = xyz$ it holds that $y \in aa^\star$ and $l = |y|$ with $0 < l \leq p$. This implies $xy^qz = a^{(l+1)\cdot q}$. Because $l \geq 1$, it holds that $(l + 1) \geq 2$. $(l + 1) \cdot q$ thus has at least one divider greater than 1. It is impossible the divider is $(l + 1) \cdot q$ because q is a prime number, therefore $q > 1$. We conclude $(l + 1) \cdot q$ has at least one[1] divider not equal to 1 or itself. Therefore $xy^qz \notin L$, regardless of the way we break down s and L is not regular. □

4. $L = \{0^m1^n \in \Sigma_1^* \mid m \neq n\}$. Here we first consider the language $L' = L^c \cap \{0^\star1^\star\} = \{0^m1^n \in \Sigma_1^* \mid m = n\}$

   Proof. Consider the string 0^p1^p with p the unknown pumping length. For every breakdown $s = xyz$ it holds that $y \in 00^\star$ and $l = |y|$ with $0 < l \leq p$. Therefore $xy^2z = 0^{p+l}1^p \notin L'$, regardless of the choices for x, y and z. Thus L' is not regular. □

   Since the intersection is closed for regular languages, if the intersection is not regular, it implies at least one of the operands is not regular. Since the second operand is defined using a regular expression, we know for sure the first operand L^c is not regular. Since the complement is closed for regular languages as well, L is not regular.

5. $L = \{w \in \Sigma_1^* \mid w \text{ is not a palindrome}\}$. We consider $L' = L^c = \{w \in \Sigma_1^* \mid w \text{ is a palindrome}\}$.

   Proof. Consider the string 0^p10^p with p the unknown pumping length. For every valid breakdown $s = xyz$ it holds that $y \in 00^\star$ and $l = |y|$ with $0 < l \leq p$. Therefore $xy^2z = 0^{p+l}10^p \notin L'$, thus $L'$is not regular. □

   Because the complement is closed for regular languages, L is not regular as well.

6. $L = \{a^ib^jc^k \in \Sigma_3^* \mid i, j, k \geq 0 \text{ and given } i = 1, \text{ then } j = k\}$. We consider $L' = L \cap \{ab^\star c^\star\} = \{ab^nc^n \in \Sigma_3^*\}$

   Proof. Consider the string $s = ab^pc^p$ with p the unknown pumping length. There exist two possible ways to subdivide the string into x, y and z:

   (a) $x = \epsilon$ and $y \in ab^\star$ with $l = |y|$. In this case $xy^2z \in L'' = ab^\star ab^{p+l-1}c^p$. Since $L \cap L'' = \emptyset$, it is impossible to "pump" such strings.

   (b) $x \in ab^\star$ and $y \in bb^\star$ with $l = |y|$ and $0 < l \leq p$, then $xy^2z = ab^{p+l}c^p$. In that case the equivalent number of b's and c's doesn't hold anymore. This configuration is not possible as well.

   With this case analysis we conclude it is impossible to "pump" the string, regardless of the choices of x, y and z. Therefore L' is not regular. □

   Because the intersection is closed under regular expressions, and the second operand is regular, the first operand is not regular.

◇

[1] It is possible that $l + 1 = q$, in that case, there is only one divider.