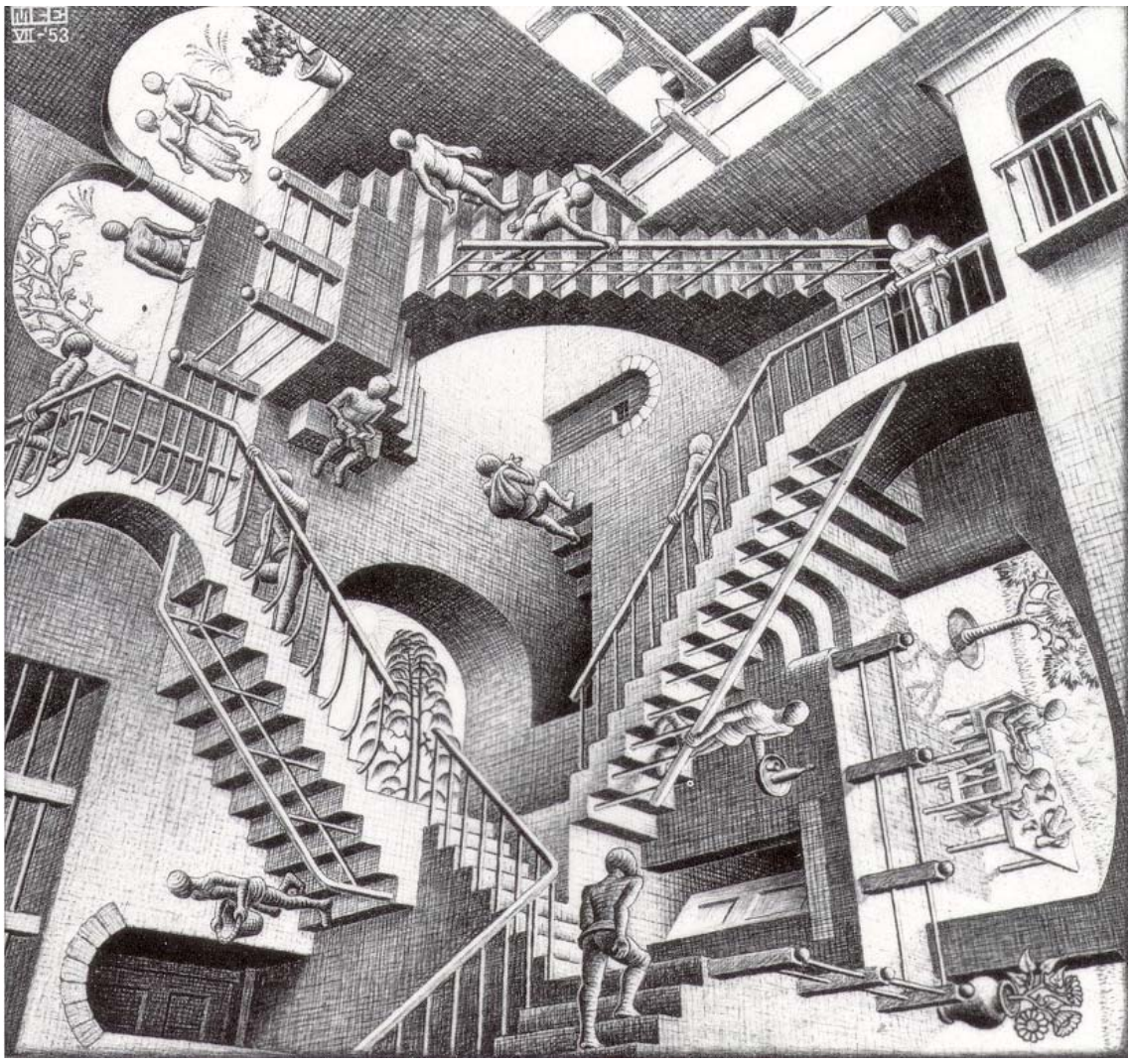
FORMAL LANGUAGES, AUTOMATA AND THEORY OF COMPUTATION



EXERCISES ON REGULAR LANGUAGES

2010

Introduction

This compendium contains exercises about regular languages for the course *Formal Languages, Automata and Theory of Computation* at the School of Innovation, Design and Technology, Mälardalen University. The notation used throughout this compendium is taken from the Swedish course book [1], and is described on the next page. All exercises has a solution in the end of the compendium, including those marked with ☞; Those marked with ☞; are test assignments.

The compendium is outlined as follows. There are five sections; *Mathematical preliminaries*, *Regular expressions*, *Finite Automata*, *Regular grammar* and *non-regularity*. Each section starts with some simple exercises intended to learn new concepts. The difficulty of the exercises then increases, and in the end there are test assignments.

Study Technique

- Discuss your solutions in the group! The other students are your biggest support in this course, use that resource!
- Try to solve the easy exercises in the beginning of each section!
- Don't look at the solution until you have tried to solve an exercise. If you cannot solve it, look at the solution and try to grasp the basic idea of the solution and try to solve it again. You won't learn anything from just copying a solution!
- To ensure that a property holds for a certain automation, grammar, expression etc. it might help to elaborate with different strings. Try to construct strings which are violating what you are trying to prove!

Good luck with the course!

[1] Lennart Salling, Formella språk, automater och beräkningar, 2001

Table of Content

Notation

Notation	*Meaning*	*Example*	*Other notation*
a	The character 'a'	a,b,c	a, b, c
foo	The string "foo"	bar, thing	*bar*, *foo*
w, x, y, z	Arbitrary string	$w \in \Sigma^*$	
ε	The empty string		λ
k	Numerical constant	$k, -3, 1, 42$	
n, m	Numerical variable	$n = 2m$	
L, S, A, B	Language, set	$L = \Sigma^*$	
$\emptyset$	The empty set		$\{\}$
Σ	An alphabet	$\Sigma = \{\mathtt{a}, \mathtt{b}\}$	
$\{w \mid \textit{foo}\}$	The set of all w such that *foo* holds	$\{w \mid w^{\textit{rev}} = w\}$	
$\mathtt{a}^k$	k repetitions of the character a	$\mathtt{a}^4 = \mathtt{aaaa}$	a^k, a^4
w^k	k repetitions of the string w	$w^3 = www$	
A^k	The set of all strings with k arbitrary elements from the set A	$\{1, 2\}^2 = \{11, 12, 21, 22\}$	
A^*	Kleene star operation on A	$\{0, 1\}^* = \{\varepsilon, 0, 1, 00, \ldots\}$, $\mathtt{a}^* = \{\varepsilon, \mathtt{a}, \mathtt{aa}, \ldots\}$, $(\mathtt{foo})^* = \{\varepsilon, \mathtt{foo}, \mathtt{foofoo}, \ldots\}$	
A^+	Plus operator on the set A	$\{0, 1\}^+ = \{0, 1, 00, 01, 10, \ldots\}$	
$w^{\textit{rev}}$	Reverse of the string w	$(\mathtt{foobar})^{\textit{rev}} = \mathtt{raboof}$	w^R
$\overline{L}$	The complement language to L		$\neg(L), L^C$
2^A	The powerset of A	$2^{\{0,1\}} = \{\emptyset, \{0\}, \{1\}, \{0, 1\}\}$	
$\lvert A \rvert$	Number of elements in A (sometimes denoted *cardinality*)	$\lvert\{1, 2, 3\}\rvert = 3, \lvert\mathtt{automation}\rvert = 7$	
$\lvert w \rvert_a$	The number of a's in the string w	$\lvert\mathtt{abracadabra}\rvert_{\mathtt{a}} = 5$	$n_a(w), \#_a(w)$
$\textit{prefix}(w)$	The set of strings x such that $w = xz$	$\textit{prefix}(\mathtt{abc}) = \{\varepsilon, \mathtt{a}, \mathtt{ab}, \mathtt{abc}\}$	
$\textit{pprefix}(w)$	The set of proper prefixes to w	$\textit{prefix}(\mathtt{abc}) = \{\mathtt{a}, \mathtt{ab}\}$	
$\textit{prefix}(L)$	The set of strings w such that w is a prefix in some string in L	$\textit{prefix}(\Sigma^*) = \Sigma^*$	
$\textit{suffix}(w)$	The set of strings x such that $w = zx$	$\textit{suffix}(\mathtt{abc}) = \{\varepsilon, \mathtt{c}, \mathtt{bc}, \mathtt{abc}\}$	
$\textit{psuffix}(w)$	The set of proper suffixes to w	$\textit{psuffix}(\mathtt{abc}) = \{\mathtt{c}, \mathtt{bc}\}$	
$\textit{suffix}(L)$	The set of strings w such that w is a suffix in some string in L	$\textit{suffix}(\Sigma^*) = \Sigma^*$	
$A \cup B$	The union of A and B	$\{1, 2, 3\} \cup \{2, 3, 4\} = \{1, 2, 3, 4\}$	$A + B$
$A \cap B$	The intersection of A and B	$\{1, 2, 3\} \cap \{2, 3, 4\} = \{2, 3\}$	
xy	The concatenation of the strings x and y	$x = \mathtt{foo}, y = \mathtt{bar}, xy = \mathtt{foobar}$	
$A - B$	The set of all elements which are in A but not in B	$\{1, 2, 3\} - \{2, 3, 4\} = \{1\}$	$A \setminus B$
$A \times B$	The set of all combinations of an element in A concatenated with an element in B.	$\{\mathtt{a}, \mathtt{b}\} \times \{\mathtt{c}, \mathtt{d}\} = \{\mathtt{ac}, \mathtt{ad}, \mathtt{bc}, \mathtt{bd}\}$	
$\#$	The end of a string/stack		
$\mathbb{N}$	The set of natural numbers	$\mathbb{N} = \{0, 1, 2, \ldots\}$	
$\mathbb{Z}$	The set of integers	$\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$	
$\mathbb{R}$	The set of real numbers	$47,011 \in \mathbb{R}$	
$\therefore$	Ergo, hence, therefore		
$\square$	End of proof		■

1 Mathematical Preliminaries

Exercise 1.1. (o1)

Construct the powerset for the following sets.

(i) $\{\mathtt{a}, \mathtt{b}\}$

(ii) $\{0, 1\} \cup \{1, 2\}$

(iii) $\{z\}$

(iv) $\{0, 1, 2, 3, 4\} \cap \{1, 3, 5, a\}$

(v) $\{0, 1, 2, 3\} - \{1, 3, 5, a\}$

(vi) $\emptyset$ (the empty set)

Exercise 1.2. (o2)

Determine the cardinality of the following languages over the alphabet $\Sigma = \{0, 1\}$ (That is, are they finite, infinite and countable, or infinite and uncountable). Prove your answers.

(i) Σ^0

(ii) Σ^4

(iii) $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \cdots$

(iv) 2^Σ

(v) 2^{Σ^*}, that is the powerset of all *languages* over Σ^*. (Hint: use a diagonalisation argument!)

Exercise 1.3. (o3)

Find a possible alphabet Σ for the following languages. A word `foobar` should be interpreted as a string of characters `f`, `o`, `o`, `b` , `a` and `r`.

(i) The language $L = \{\mathtt{oh}, \mathtt{ouch}, \mathtt{ugh}\}$

(ii) The language $L = \{\mathtt{apple}, \mathtt{pear}, \mathtt{4711}\}$

(iii) The language of all binary strings

Exercise 1.4. (o4)

Describe what the Kleene star operation * over the following alphabets produces.

(i) $\Sigma = \{0, 1\}$

(ii) $\Sigma = \{\mathtt{a}\}$

(iii) $\Sigma = \emptyset$ (the empty alphabet)

Exercise 1.5. (o5)

Describe what the $^+$-operation over the following alphabets produces.

(i) $\Sigma = \{0, 1\}$

(ii) $\Sigma = \{\mathtt{a}\}$

(iii) $\Sigma = \emptyset$ (the empty alphabet)

Exercise 1.6. (o6)

State the alphabet Σ for the following languages :

(i) $L = \Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \ldots\}$

(ii) $L = \Sigma^+ = \{\mathtt{a}, \mathtt{aa}, \mathtt{aaa}, \ldots\}$

(iii) $L = \Sigma^+ = \{\varepsilon\}$

Exercise 1.7. (o7)

Assuming that $\Sigma = \{0, 1\}$, construct complement languages for the following.

(i) $\overline{\{010, 101, 11\}}$

(ii) $\overline{\Sigma^* - \{110\}}$

(iii) $\overline{\Sigma^+ - \varepsilon}$

Exercise 1.8. (o8)

State the following languages explicitly.

(i) $2^{\{\mathtt{a},\mathtt{b}\}} - 2^{\{\mathtt{a},\mathtt{c}\}}$

(ii) $\{\mathtt{a}, \mathtt{b}\} \times \{1, 2, 3\} \times \emptyset$

(iii) $\{x \mid x \in \mathbb{N} \wedge \exists y \in \mathbb{N} : y < 10 \wedge (y + 2 = x)\}$ ($\mathbb{N}$ is the set of all non-negative integers)

Exercise 1.9. (o9)

Let Σ be an alphabet, and let ε be the empty string over Σ.

(i) is ε in Σ?

(ii) does it hold that $\varepsilon\varepsilon\varepsilon = \varepsilon$? does it hold for ε^i where $i \geq 2$?

(iii) Let x and y be two strings over Σ. Is the concatenation of x and y always the same as the concatenation of y and x?

Exercise 1.10. (o10)

A string is infinite when its length is infinite. Let Σ be an arbitrary alphabet.

(i) Does Σ^* contain any infinite string?

(ii) If Σ would be an *infinite* alphabet (which it actually may not be), would the answer still be the same?

Exercise 1.11. (o74)

Prove that $6^n \equiv 0 \pmod 9$ for all integers $n \geq 2$.

Exercise 1.12. (o75)

Prove by induction that $|S^2| = 2^{|S|}$ (that is, that the size of the powerset is equal to 2 to the power of the size of the original set) for all finite sets S.

Exercise 1.13. (o76)

Let x be a string and let x^{rev} be " the same" string but backwards. Prove that $(xy)^{rev} = y^{rev}x^{rev}$ for arbitrary strings x, y over an alphabet Σ. (Hint: try to define x^{rev} inductively)

Exercise 1.14. (o12)

Prove by induction that $|A \times B| = |A| \cdot |B|$ for all sets A and B.

Exercise 1.15. (o13)

Prove that $2^0 + 2^1 + 2^2 + \cdots + 2^n = 2^{n+1} - 1$ for all integers $n \geq 0$.

Solutions

Solution 1.1. (o1)

(i) $2^S = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$

(ii) $S = \{0, 1, 2\}, 2^S = \{\emptyset, \{0\}, \{1\}, \{2\}, \{0, 1\}, \{0, 2\}, \{1, 2\}, \{0, 1, 2\}\}$

(iii) $2^S = \{\emptyset, \{\mathtt{z}\}\}$

(iv) $S = \{1, 3\}, 2^S = \{\emptyset, \{1\}, \{3\}, \{1, 3\}\}$

(v) $S = \{0, 2\}, 2^S = \{\emptyset, \{0\}, \{2\}, \{0, 2\}\}$

(vi) $2^S = \{\emptyset\}$

Solution 1.2. (o2)

(i) $\Sigma^0 = \{\varepsilon\}$ $\therefore$ *The set is finite.*

(ii)

$$\Sigma^4 = \{ \quad 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, \\ 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111 \quad \}$$

$\therefore$ *The set is finite.*

(iii) Each set in the infinite and countable series is finite and may therefore be counted by mapping it to the natural numbers. Thereby, the full set can be counted. $\therefore$ The set is infinite and countable.

(iv) $2^\Sigma = \{ \{\varepsilon\}, \{0\}, \{1\}, \{0, 1\} \}$ $\therefore$ *The set is finite.*

(v) $\therefore$ The set is infinite and uncountable.

Solution 1.3. (o3)

(i) $\Sigma = \{\mathtt{o}, \mathtt{h}, \mathtt{u}, \mathtt{c}, \mathtt{g}\}$

(ii) $\Sigma = \{\mathtt{a}, \mathtt{p}, \mathtt{l}, \mathtt{e}, \mathtt{r}, \mathtt{a}, \mathtt{4}, \mathtt{7}, \mathtt{1}\}$

(iii) $\Sigma = \{0, 1\}$

Solution 1.4. (o4)

(i) All binary strings.

(ii) All strings which contains nothing but `a`'s (including the empty string).

(iii) The language which contains only the empty string.

Solution 1.5. (o5)

(i) Binary strings containing at least one character.

(ii) Strings containing only `a`'s, with at least one character.

(iii) The language which contains only the empty string.

Solution 1.6. (o6)

(i) $\Sigma = \{0, 1\}$

(ii) $\Sigma = \{\mathtt{a}\}$

(iii) $\Sigma = \emptyset$

Solution 1.7. (o7)

(i) $L = \{\varepsilon, 0, 1, 00, 01, 10, 000, 001, 011, 100, 110, 111, 0000, \ldots\}$

(ii) $L = \{110\}$

(iii) $L = \{\varepsilon\}$

Solution 1.8. (o8)

(i) $\big\{\{\mathtt{b}\}, \{\mathtt{a}, \mathtt{b}\}\big\}$

(ii) $\emptyset$

(iii) $\{2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$

Solution 1.9. (o9)

(i) No.

(ii) Yes. Yes.

(iii) No, consider for instance $\Sigma = \{\mathtt{a}, \mathtt{b}\}, x = \mathtt{a}, y = \mathtt{b}, xy = \mathtt{ab}, yx = \mathtt{ba}$.

Solution 1.10. (o10)

(i) No, even though the language is infinite, all strings have finite length.

(ii) No, each symbol in the alphabet still has finite length, and each string has finite length.

Solution 1.11. (o74)

Claim: $6^n \equiv 0 \pmod 9$ for all integers $n \geq 2$.

Verification:

$$6^2 = 36 = 9 \cdot \underbrace{4}_{\text{heltal}}, \quad 6^3 = 216 = 9 \cdot \underbrace{24}_{\text{heltal}}, \quad 6^4 = 1296 = 9 \cdot \underbrace{144}_{\text{heltal}}$$

Base step: We choose 6^2, that is $n = 2$ as base case, since this is the least n for which the claim must hold. Since $6^2 = 36 = 9 \cdot 4$, the claim holds for the base case.

Induction step: We assume that the claim is true for a certain case and try to show that it is true for the following case as well.

Thus, assume that 9 divides 6^p, that is $6^p = 9k$ for some integer k. We want to show that this implies that $6^{p+1} = 9\ell$ for some integer ℓ.

If $6^p = 9k$ for some p and some integer k, then the following holds.

$$\begin{aligned} 6^{p+1} &= 6 \cdot 6^p \\ &= 6 \cdot 9k \\ &= 9 \underbrace{(6k)}_{\text{integer}} \\ &= 9\ell \text{ where } \ell \text{ is an integer} \end{aligned}$$

Summary: The claim holds for $n = 2$, and if it holds for some case p, then it also holds for the case $p + 1$. Thus, it holds for all cases, which concludes the proof. □

Solution 1.12. (o75)

Claim: $|S^2| = 2^{|S|}$ for all finite sets S.

Verification:

$$\begin{aligned} |\emptyset^2| = 1 &= 2^0 = 2^{|\emptyset|} \\ |\{x\}^2| = 2 &= 2^1 = 2^{|\{x\}|} \\ |\{x, y\}^2| = 4 &= 2^2 = 2^{|\{x,y\}|} \end{aligned}$$

Base step: We choose $S = \emptyset$ as base step, since this is the least set for which the claim should hold. According to the verification above, the claim should hold for the base step.

Induction step: We assume that the claim is true for a certain case, and try to prove it for the next case.

Thus, assume that $|S^2| = 2^{|S|}$ for some set S. We want to show that this implies that $|(S \cup \{x\})^2| = 2^{|S \cup \{x\}|}$ for an arbitrary element $x \notin S$.

If $|S^2| = 2^{|S|}$ for some set S, then the following holds.

$$\begin{aligned} (S \cup \{x\})^2 &= S^2 \cup T \text{ where } T = \{t \cup \{x\} \mid t \in S^2\} \\ |(S \cup \{x\})^2| &= |S^2 \cup T| \\ &= |S^2| + |T| \text{ since } S^2 \text{ and } T \text{ are disjoint } (x \notin S \rightarrow x \notin 2^S) \\ &= |S^2| + |\{t \cup \{x\} \mid t \in S^2\}| \\ &= |S^2| + |S^2| \\ &= 2 \cdot |S^2| \\ &= 2 \cdot 2^{|S|} \text{ (from the inductive hypothesis)} \\ &= 2^{|S|+1} \\ &= 2^{|S \cup \{x\}|} \text{ since } x \notin S \end{aligned}$$

Summary: The claim holds for $S = \emptyset$, and if the claim holds for some set S then it also holds for the set $S \cup \{x\}$ where $x \notin S$. Thus, it holds in all cases and the proof is concluded. □

Solution 1.13. (o76)

$\varepsilon^{rev} = \varepsilon$ for the empty string ε. $a^{rev} = a$ for a string with containing only a single character. A non-empty string x can be partitioned into two substrings $x = ay$ where a contains one or zero characters. We can then define $(ay)^{rev} = y^{rev}a$ where $|a| \leq 1$.

Now we will show that $(xy)^{rev} = y^{rev}x^{rev}$. First assume that x is the empty string. If so, the claim is true, since $(\varepsilon y)^{rev} = y^{rev} = y^{rev}\varepsilon$. Now assume that x is a string a containing only one character. If so, the claim is true as well, since $(ay)^{rev} = y^{rev}a$ by definition. These are our base steps.

Now we assume that the claim $(uv)^{rev} = v^{rev}u^{rev}$ holds for two strings u, v. This is our inductive hypothesis. Now we want to show that $(auv)^{rev} = v^{rev}u^{rev}a$.

$(auv)^{rev} = (uv)^{rev}a$ by definition. Furthermore, $(uv)^{rev} = v^{rev}u^{rev}$ by inductive hypothesis, which yields $(uv)^{rev}a = v^{rev}u^{rev}a$, which is what we wanted to show.

Since we have shown that the claim holds for a (two) base cases, and that it holds for any string which may be constructed by concatenating a character to the beginning of the first string, the claim holds for all strings. □

Solution 1.14. (o12)

$|A \times B| = |A| \cdot |B|$ for the empty string $A = \emptyset$ and an arbitrary set B. The claim also holds for $A = \emptyset$ and an arbitrary B. Now assume that the claim holds for two sets A and B (the induction hypothesis). We want to show that $|A \cup \{x\} \times B| = |A \cup \{x\}| \cdot |B|$ for an arbitrary element $x \notin A$ (if $x \in A$ the claim holds trivially, since $x \in A \rightarrow A \cup \{x\} = A$).

We start with the left hand side. The cross product implies that $A \cup \{x\} \times B = A \times B \cup \{x\} \times B$. Since $A \times B$ and $\{x\} \times B$ are disjoint (since $x \notin A$), it holds that $|A \times B \cup \{x\} \times B| = |A \times B| + |\{x\} \times B| = |A \times B| + |B|$.

For the right hand side it holds that $|A \cup \{x\}| = |A| + 1$. Thus, we have $|A \cup \{x\}| \cdot |B| = (|A|+1) \cdot |B| = |A| \cdot |B| + |B| = |A \times B| + |B|$. But then it also holds that $|A \cup \{x\} \times B = |A \cup \{x\}| \cdot |B|$, which concludes the proof. □

Solution 1.15. (o13)

Claim: $2^0 + 2^1 + 2^2 + \cdots + 2^n = 2^{n+1} - 1$ for all integers $n \geq 0$.

Verification:

$$\begin{aligned} 2^0 = 1 &= 2^1 - 1 = 1 \\ 2^0 + 2^1 = 3 &= 2^2 - 1 = 3 \\ 2^0 + 2^1 + 2^2 = 7 &= 2^3 - 1 = 7 \end{aligned}$$

Base step: We choose $n = 0$ as base case, since this is the least number for which the claim should hold. According to the verification above, the claim holds for the base step.

Induction step: We assume that the claim holds for a certain case, and try to prove that it holds also for the following case.

Thus, assume that $2^0 + 2^1 + 2^2 + \cdots + 2^k = 2^{k+1} - 1$ for some number k. We want to show that this implies that $2^0 + 2^1 + 2^2 + \cdots + 2^k + 2^{k+1} = 2^{k+2} - 1$.

If $2^0 + 2^1 + 2^2 + \cdots + 2^k = 2^{k+1} - 1$ for some number k, then the following holds.

$$\begin{aligned} 2^0 + 2^1 + 2^2 + \cdots + 2^k + 2^{k+1} &= (2^{k+1} - 1) + 2^{k+1} && \textit{(from the inductive hypothesis)} \\ &= 2^{k+1} + 2^{k+1} - 1 \\ &= 2 \cdot 2^{k+1} - 1 \\ &= 2^{k+2} - 1 \end{aligned}$$

Summary: The claim holds for the base case $k = 0$, and if it holds for a number k then it also holds for the following number $k + 1$. Thus it holds for all cases, which concludes the proof. □

2 Regular expressions

Exercise 2.1. (o23)

Construct regular expressions representing languages, over the alphabet $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$, in which for every string w it holds that:

(i) The number of a's in w is even.

(ii) There are $4i + 1$ b's in w. ($i \geq 0$)

(iii) $|w| = 3i$. ($i \geq 0$)

Exercise 2.2. (o24)

A regular expression is *ambiguous* when there exists a string which can be constructed in two different ways from the regular expression. Which of the following regular expressions are ambiguous?

(i) $\mathtt{a}\big((\mathtt{ab})^*\mathtt{cd}\big)^* \cup \mathtt{a}(\mathtt{ababcb}^*)^*\mathtt{a}^*$

(ii) $\mathtt{aab}^*(\mathtt{ab})^* \cup \mathtt{ab}^* \cup \mathtt{a}^*\mathtt{bba}^*$

(iii) $\mathtt{aaba}^* \cup \mathtt{aaaba} \cup \mathtt{aabba}^* \cup \mathtt{a}$

Exercise 2.3. (o40)

Consider the regular expression $\big(a(cd)^*b\big)^*$.

(i) Find a string over $\{a, b, c, d\}^4$ which matches the expression.

(ii) Find a string over $\{a, b, c, d\}^4$ which does not match the expression.

Exercise 2.4. (o41)

Find a regular expression for the language consisting of alternating zeroes and ones.

Exercise 2.5. (o42)

Find a regular expression for the language L over $\Sigma = \{a, b\}$ consisting of strings which contain exactly two or exactly three b's.

Exercise 2.6. (o43)

Find a regular expression over the language L over the alphabet $\Sigma = \{a, b\}$ consisting of strings where the number of b's can be evenly divided by 3.

Exercise 2.7. (o44)

Describe which languages the following regular expressions represents, using common english.

(i) $(0 \cup 1)^*01$

(ii) 1^*01^*

(iii) $(11)^*$

(iv) $(0^*10^*10^*)^*$

(v) $(0 \cup 1)^*01(0 \cup 1)^*$

(vi) 1^*0^*

(vii) $(10 \cup 0)^*(1 \cup 10)^*$

(viii) $0^*(1 \cup 000^*)^*0^*$

Exercise 2.8. (o22)

Let E be the following regular expression:

$$c^*\big(\varepsilon \cup \mathtt{a}(\mathtt{a} \cup \mathtt{b} \cup \mathtt{c})^* \cup (\mathtt{a} \cup \mathtt{b} \cup \mathtt{c})^*\mathtt{b}\big)\mathtt{c}^*$$

(i) is $L(E) = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^*$?

(ii) is $L(EE) = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^*$?

☞ TEST ASSIGNMENT 1. (LESSON) (o45)

Find a regular expression which represents the set of strings over $\{a, b\}$ which contains the two substrings aa and bb.

Solutions

Solution 2.1. (o23)

(i) $(\mathtt{b} \cup \mathtt{c})^* \Big(\big(\mathtt{a} (\mathtt{b} \cup \mathtt{c})^* \big)^2 \Big)^*$

(ii) $(\mathtt{a} \cup \mathtt{c})^* \mathtt{b} (\mathtt{a} \cup \mathtt{c})^* \Big(\big(\mathtt{b} (\mathtt{a} \cup \mathtt{c})^* \big)^4 \Big)^*$

(iii) $(\Sigma^3)^*$

Solution 2.2. (o24)

(i) Ambiguous. For instance, the string a can be constructed by using $\mathtt{a} \big((\mathtt{ab})^* \mathtt{cd} \big)^*$ or $\mathtt{a} (\mathtt{ababcb}^*)^* \mathtt{a}^*$.

(ii) Ambiguous. The string abb can be constructed either by $\mathtt{ab}^*$ or $\mathtt{a}^* \mathtt{bba}^*$.

(iii) Unambiguous.

Solution 2.3. (o40)

(i) abab matches.

(ii) abcd doesn't match, since every c must be preceded by an a in order for the string to match.

Solution 2.4. (o41)

Alternative 1:

1. $(01)^*$ is the language of zero or more 01.
2. $(1 \cup \varepsilon)(01)^*$ is the language of alternating zeroes and ones which ends in 1.
3. $(1 \cup \varepsilon)(01)^*(0 + \varepsilon)$ is the language of alternating zeroes and ones.

Alternative 2: $(0 \cup \varepsilon)(10)^*(1 + \varepsilon)$ according to the above.

Solution 2.5. (o42)

1. a^*ba^* is the set of strings with exactly one b.
2. $a^*ba^*ba^*$ is the set of strings with exactly two b's.
3. $a^*ba^*ba^*ba^*$ is the set of strings with exactly three b's.
4. The language L is then $a^*ba^*ba^* \cup a^*ba^*ba^*ba^*$.

Solution 2.6. (o43)

1. $a^*ba^*ba^*ba^*$ are all strings with exactly three b's.
2. $(a^*ba^*ba^*ba^*)^*a^*$ is then the language L.

Solution 2.7. (o44)

(i) An arbitrary number of binary characters (0 or 1) precedes the substring 01.

(ii) Strings that must contain a zero but which otherwise consists only of ones.

(iii) Strings consisting only of ones and which lengths are even.

(iv) An arbitrary number of repetitions of a string consisting two 1's and an arbitrary number of zeroes in arbitrary positions.

(v) Strings containing the substring 01.

(vi) Strings on the form $111\ldots000\ldots$, that is, strings that begins with zero or more ones followed by zero or more zeroes.

(vii)
- $(10 \cup 0)^*$ is all strings which doesn't contain the substring 11.
- $(1 \cup 10)^*$ is all strings which doesn't contain the substring 00.

so the concatenation of these is all strings where each occurrence of 00 precedes all occurrences of 11.

(viii) Al strings which doesn't contain the substring 101.

Solution 2.8. (o22)

(i) No. See for instance the string `ba` which cannot be produced by the regular expression.

(ii) Yes.

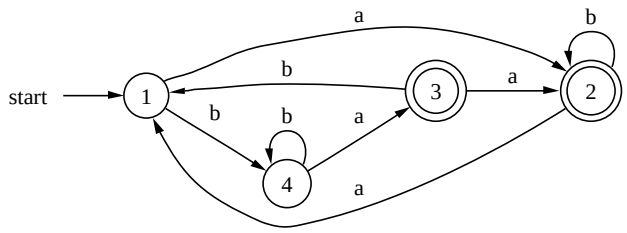
☞ SOLUTION TEST ASSIGNMENT 1. (o45)

$$(a \cup b)^* \Big(\big(aa(a \cup b)^* bb\big) \cup \big(bb(a \cup b)^* aa\big) \Big) (a \cup b)^*$$

3 Finite Automata

Exercise 3.1. (o16)

Let M be the following DFA.

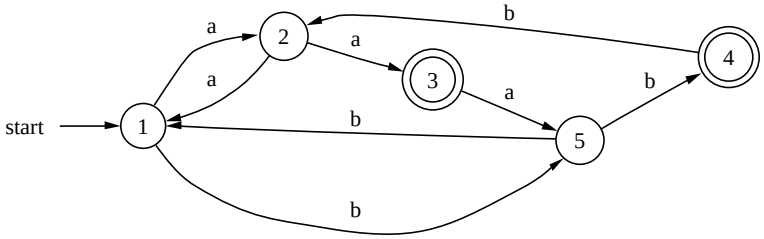


(i) Write down four string accepted by M and the sequence of configurations that shows this.

(ii) Write down four strings not accepted by M.

Exercise 3.2. (o18)

Let M be the following DFA.



(i) Write down four strings accepted by M and the sequence of configurations that shows this.

(ii) Write down four strings not accepted by M.

Exercise 3.3. (o15)

Construct a DFA which accepts the following language:

$$L = \{w \mid w \in \Sigma^* \wedge w \text{ contains the substring } 0101\}$$

That is, $w = x0101y$ for two arbitrary strings x and y.

Exercise 3.4. (o36)

Construct finite automata, deterministic of non-deterministic for the following regular expressions.

(i) $a(a \cup b)^*b$

(ii) $1(1 \cup 0)^*0 \cup 0$

(iii) 0^*10^*

(iv) $(0 \cup 1)^* 1 (0 \cup 1)^*$

(v) $0^* (1 \cup \varepsilon) 0^*$

(vi) $(0 \cup 1)(0 \cup 1)(0 \cup 1)^*$

(vii) $a(a \cup b)^* a \cup b(a \cup b)^* b \cup a \cup b$

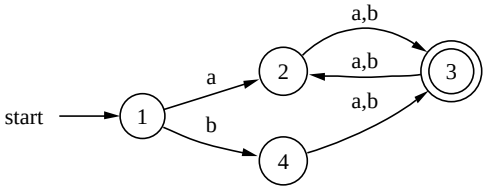
Exercise 3.5. (o37)

Formally define the automata you constructed in Exercise 3.4.

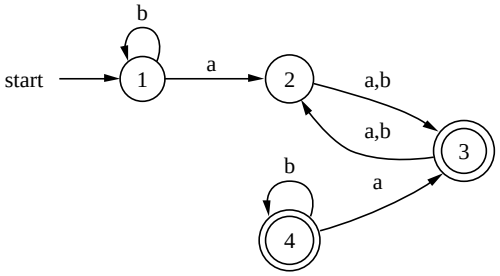
Exercise 3.6. (o17)

Which languages are accepted by the following automata:

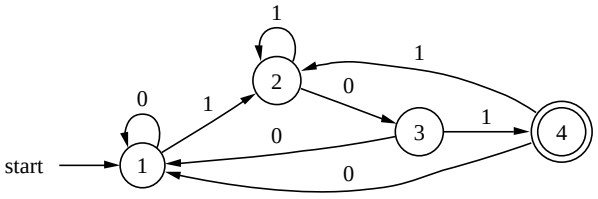
(i)



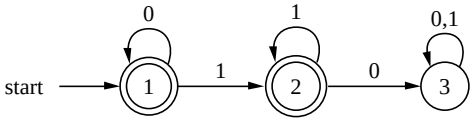
(ii)



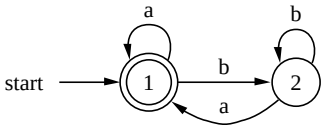
(iii)



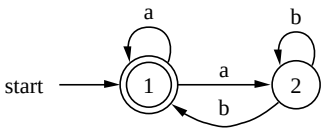
(iv)



(v)



(vi)



Exercise 3.7. (o20)

Find DFA's which accepts the following languages:

(i) Strings over $\{a, b\}$ ending in aa.

(ii) String over $\{a, b\}$ containing three consecutive a's (that is, contains the substring aaa).

(iii) Strings over $\{a, b\}$ not containing the substring aaa.

(iv) All strings over $\{a, b\}$ where each string of length 5 contains at least two a's.

(v) The set of strings over $\{a, b\}$ where each pair of a's immediately succeeds a pair of b's.

(vi) The strings $\varepsilon, 001$ and 000101.

(vii) Strings over $\{0, 1\}$ beginning with a 1 and which, if interpreted as a binary number, is a multiple of 5. As an example, the string 1010 should be accepted since $1010_2 = 10_{10}$, which is a multiple of 5. On the other hand, the string 1110 should not be accepted, since $1110_2 = 14_{10}$ which is not a multiple of 5.

(viii) $\{w \in \{a, b\}^* \mid |w|_a = 2n, |w|_b = 2m, m, n \in \mathbb{N}\}$

(ix) $\{w \in \{a, b\}^* \mid |w|_a = 3n, n \in \mathbb{N}\}$ and w doesn't contain the substring aba.

Exercise 3.8. (o25)

Construct finite automata which accepts the languages defined by the following regular expressions.

(i) $\mathtt{a}^*\mathtt{b}\mathtt{a}^*\mathtt{a}\mathtt{b}^*$

(ii) $\mathtt{b}\big((\mathtt{a}\mathtt{a}\mathtt{b}^* \cup \mathtt{a}^4)\mathtt{b}\big)^*\mathtt{a}$

(iii) $\mathtt{a}\mathtt{b}\Big(\big((\mathtt{b}\mathtt{a})^* \cup \mathtt{b}\mathtt{b}\mathtt{b}\big)^* \cup \mathtt{a}\Big)^*\mathtt{b}$

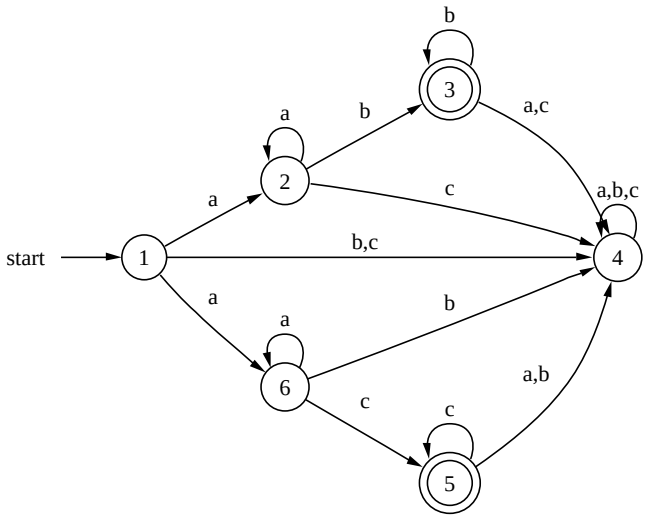
Exercise 3.9. (o29)

Construct a DFA which is equivalent to the following regular expression:

$$\big(00 \cup (1 \cup 01)(11 \cup 0)^*10\big)^*$$

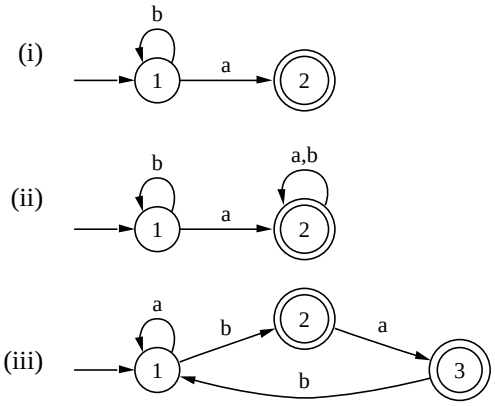
Exercise 3.10. (o19)

Prove that the following automation accepts the language $L = \{a^i b^j, a^i c^j \mid i, j \geq 1\}$.



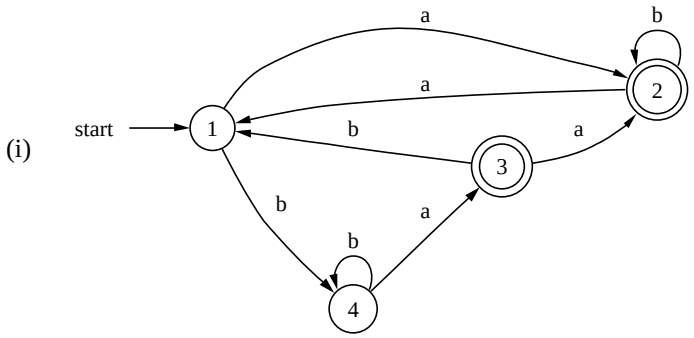
Exercise 3.11. (o35)

Convert the following finite automata to GFA's and find out which regular expressions which describes the automata's languages.

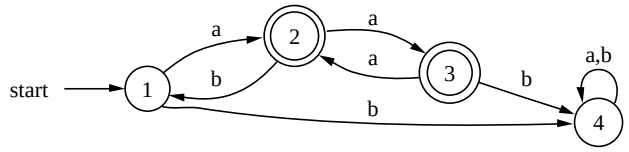


Exercise 3.12. (o27)

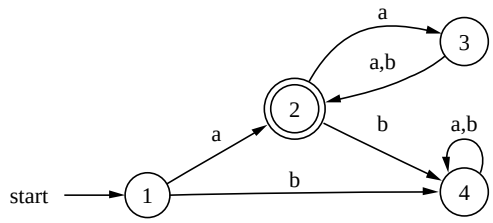
Construct regular expressions which are equivalent to the following (deterministic) finite automata:



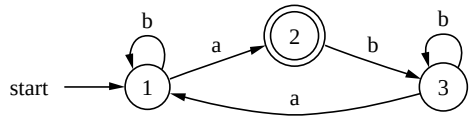
(ii)



(iii)

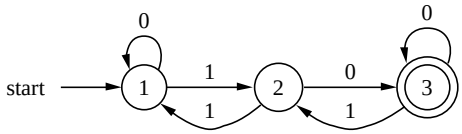


(iv)



Exercise 3.13. (○30)

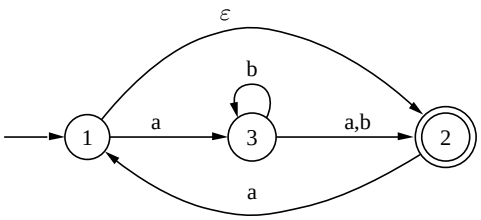
Construct a regular expression representing the following automation.



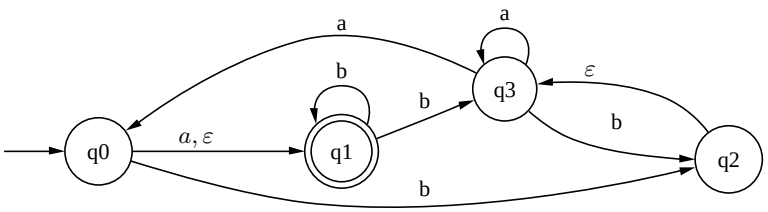
Exercise 3.14. (○38)

Convert each of the NFA's below to an equivalent DFA by subset construction.

(i)

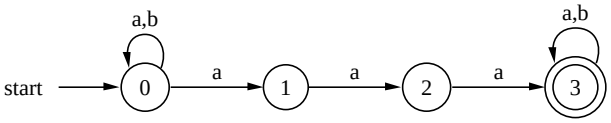


(ii)



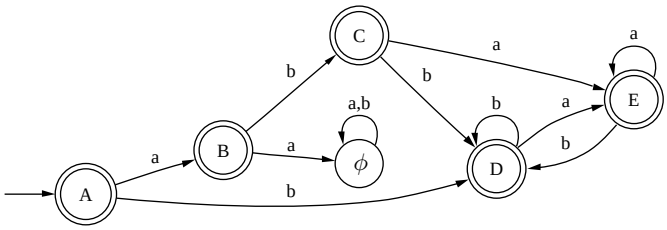
Exercise 3.15. (o65)

Convert the following NFA to a DFA.



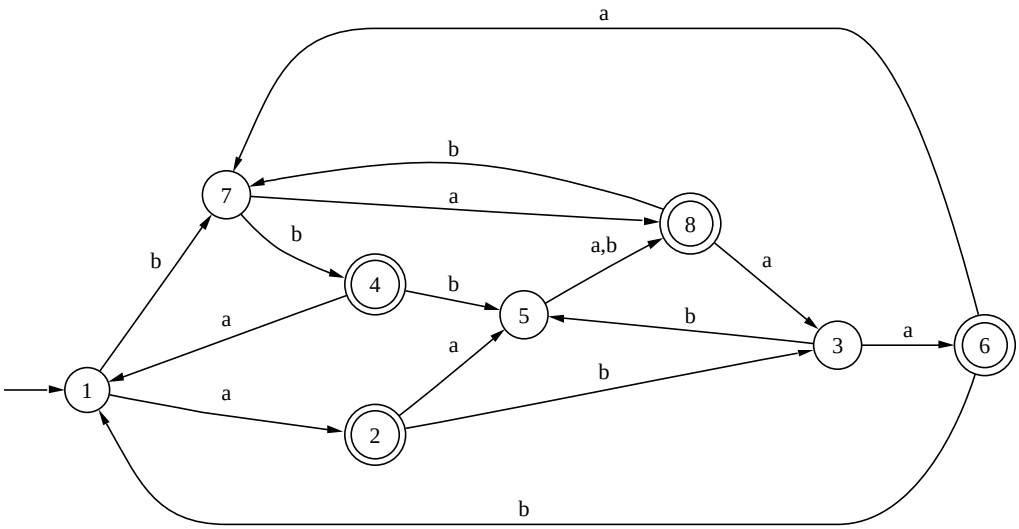
Exercise 3.16. (o39)

Minimise the DFA below by set partitioning or double complement.



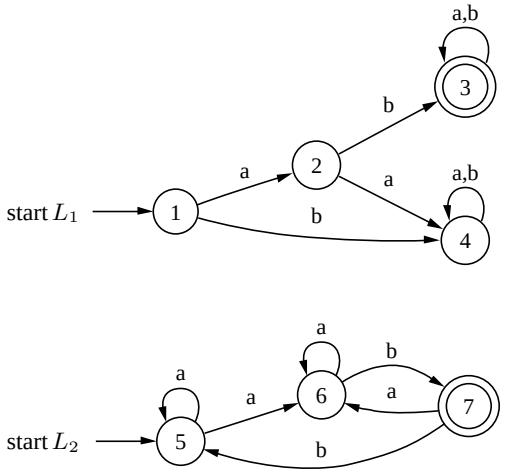
Exercise 3.17. (o80)

Minimise the following DFA and prove that your new DFA really is minimal.



Exercise 3.18. (o64)

Let L_1 and L_2 be languages represented by the following automata. Construct a minimum DFA representing $L_1 \cup L_2$.

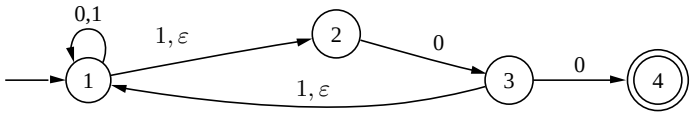


☞ TEST ASSIGNMENT 2. (LESSON) (o28)

Construct regular expressions for the automation defined in exercise 3.1.

☞ TEST ASSIGNMENT 3. (LESSON) (o11)

Convert the following NFA to an equivalent DFA by subset construction.



☞ TEST ASSIGNMENT 4. (LESSON) (o72)

Construct the minimum DFA which is equivalent with the following regular expression:

$$\big((a \cup b)^* \cup \big)(ac)^* b \cup a$$

Hint: The DFA has 8 states.

Solutions

Solution 3.1. (o16)

(i)
1. a: $\langle 1, \texttt{a}\# \rangle, \langle 2, \# \rangle$
2. bba: $\langle 1, \texttt{bba}\# \rangle, \langle 4, \texttt{ba}\# \rangle, \langle 4, \texttt{a}\# \rangle, \langle 3, \# \rangle$
3. baa: $\langle 1, \texttt{baa}\# \rangle, \langle 4, \texttt{aa}\# \rangle, \langle 3, \texttt{a}\# \rangle, \langle 2, \# \rangle$
4. baba: $\langle 1, \texttt{baba}\# \rangle, \langle 4, \texttt{aba}\# \rangle, \langle 3, \texttt{ba}\# \rangle, \langle 1, \texttt{a}\# \rangle, \langle 2, \# \rangle$

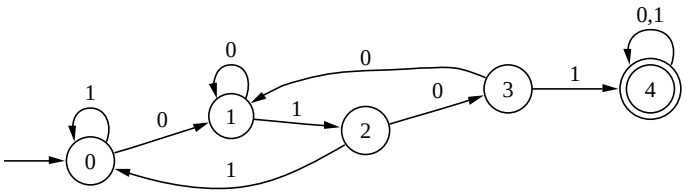
(ii)
1. ε
2. bbbb
3. aaaa
4. abab

Solution 3.2. (o18)

(i)
1. aa: $\langle 1, \texttt{aa}\# \rangle, \langle 2, \texttt{a}\# \rangle, \langle 3, \# \rangle$
2. bb: $\langle 1, \texttt{bb}\# \rangle, \langle 5, \texttt{b}\# \rangle, \langle 4, \# \rangle$
3. aaab: $\langle 1, \texttt{aaab}\# \rangle, \langle 2, \texttt{aab}\# \rangle, \langle 3, \#\texttt{ab} \rangle, \langle 5, \#\texttt{b} \rangle, \langle 4, \# \rangle$
4. bbba: $\langle 1, \texttt{bbba}\# \rangle, \langle 5, \texttt{bba}\# \rangle, \langle 4, \#\texttt{ba} \rangle, \langle 2, \#\texttt{a} \rangle, \langle 3, \# \rangle$

(ii)
1. ε
2. a
3. aaab
4. bbb

losningo15

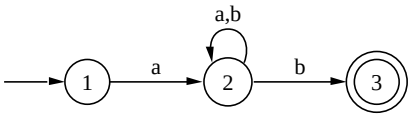


Note that if the input string w exists in the language L, we eventually will find the substring 0101 in w. We define the states corresponding to how much of the substring 0101 has been observed:

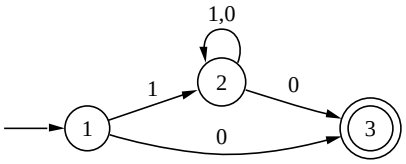
- State 0 is the initial state. In this state no part of the substring 0101 has been observed.
- In state 1, the first zero of 0101 has been observed. If a zero turns up, we continue to search for the next one in the substring 0101.
- In state 2, the substring 01 of 0101 has been observed. If the next character is a one, we return to the initial state since 011 is not a substring of 0101.
- In state 3, the string 010 has been observed. If we add another zero, we return to state 1 – this because the new zero may be the start of a new substring 0101.
- In state 4 we have observed the full substring 0101. Thus, it doesn't matter what other characters there are in the string, since we already have concluded that the string exists in the language. Consequently, state 4 is an accepting state.

Solution 3.4. (o36)

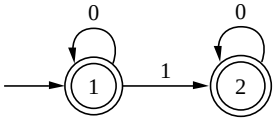
(i)



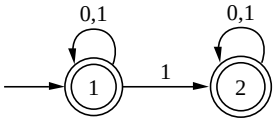
(ii)



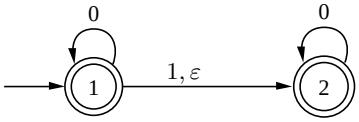
(iii)



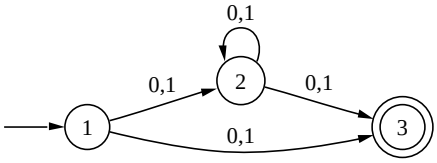
(iv)



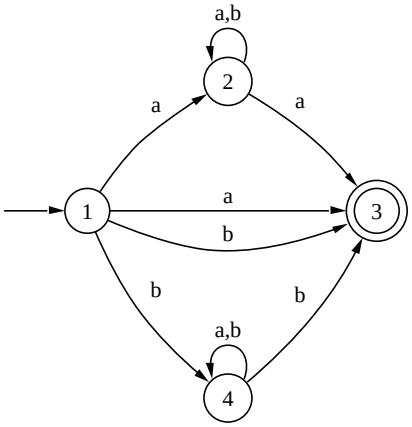
(v)



(vi)



(vii)



Solution 3.5. (o37)

(i) $M = (\{1,2,3\}, \{a,b\}, \delta, \{1\}, \{3\})$,

σ	a	b	ε
1	2		
2	2	$2,3$	
3			

(ii) $M = (\{1,2,3\}, \{0,1\}, \delta, \{1\}, \{3\})$,

σ	0	1	ε
1	3	2	
2	2, 3	2	
3			

(iii) $M = (\{1,2\}, \{0,1\}, \delta, 1, \{1,2\})$,

σ	0	1
1	1	2
2	2	

(iv) $M = (\{1,2\}, \{0,1\}, \delta, \{1\}, \{1,2\})$,

σ	0	1	ε
1	1	1, 2	
2	2	2	

(v) $M = (\{1,2\}, \{0,1\}, \delta, \{1\}, \{1,2\})$,

σ	0	1	ε
1	1	2	2
2	2		

(vi) $M = (\{1,2,3\}, \{a,b\}, \delta, \{1\}, \{3\})$,

σ	a	b	ε
1	2, 3	2, 3	
2	2, 3	2, 3	
3			

(vii) $M = (\{1,2,3,4\}, \{a,b\}, \delta, \{1\}, \{3\})$,

σ	a	b	ε
1	2, 3	3, 4	
2	2, 3	2	
3			
4	4	3, 4	

Solution 3.6. (o17)

(i) The strings with an even number of characters and length of at least 2.

(ii) The strings which contains at least one `a`, followed by an odd number of characters.

(iii) The strings that ends in 101.**NOTE: or?**

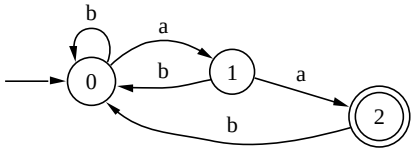
(iv) The strings on the form $0^n1^m, n, m \geq 0$.

(v) The strings that ends in `a`.

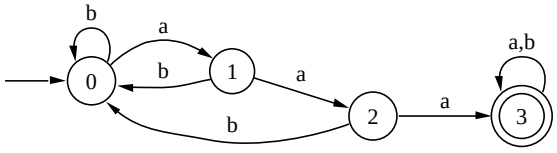
(vi) The strings that doesn't start with a `b`.

Solution 3.7. (o20)

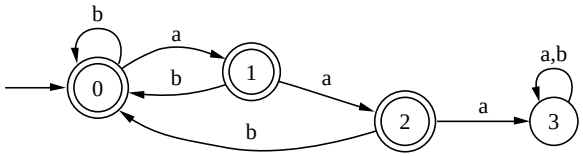
(i)



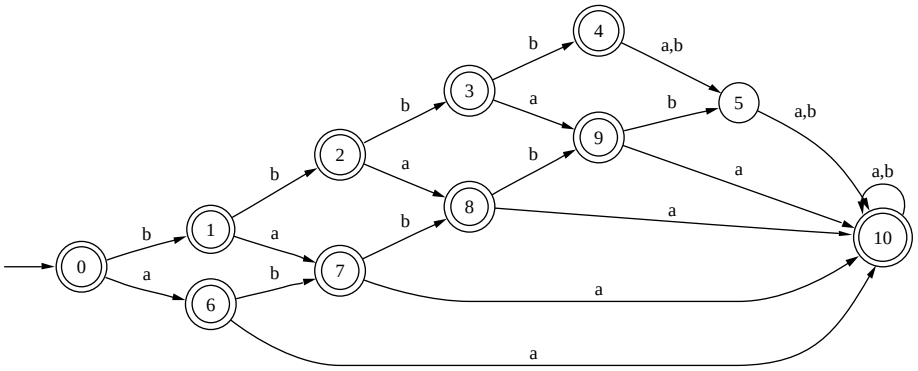
(ii)



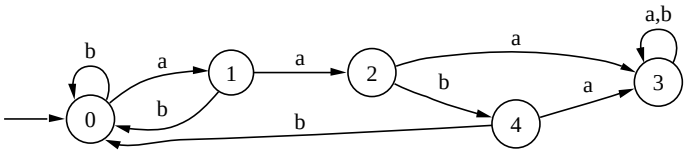
(iii)



(iv)

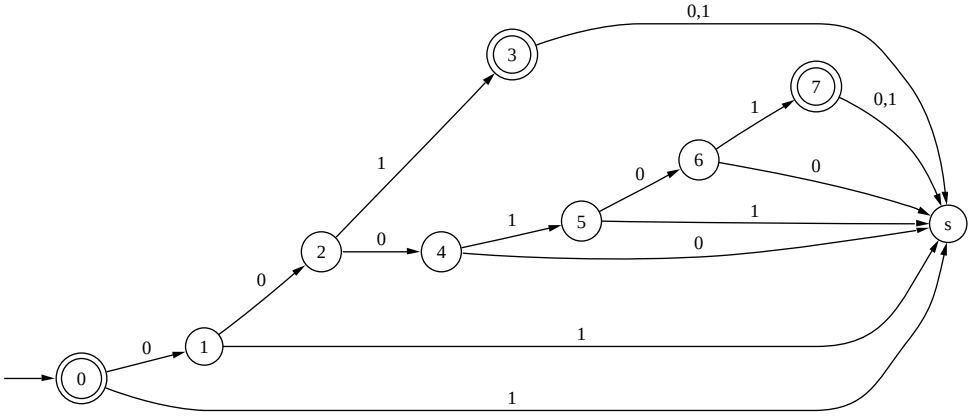


(v)

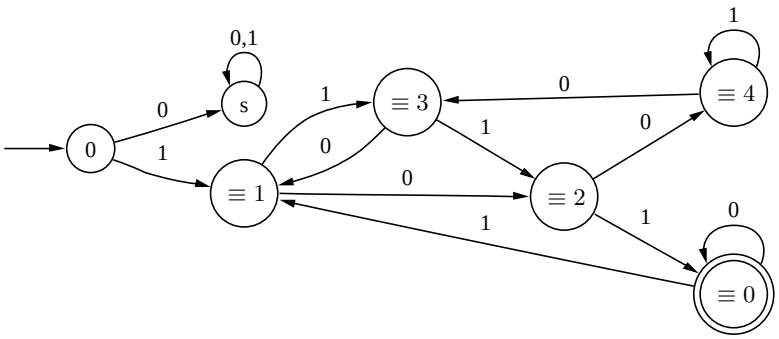


Note: state 0 should be an accepting state here.

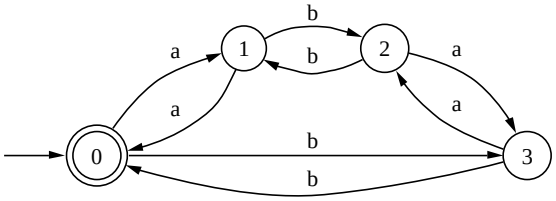
(vi)



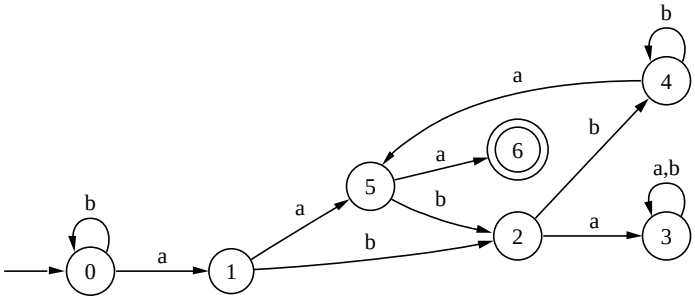
(vii)



(viii)

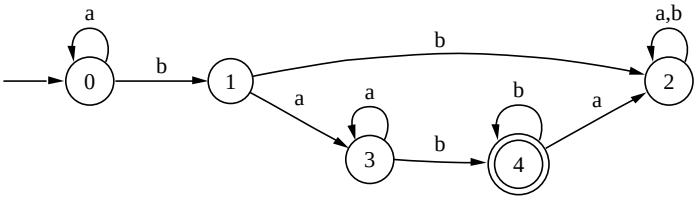


(ix)

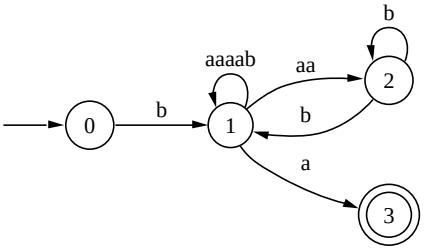


Solution 3.8. (○25)

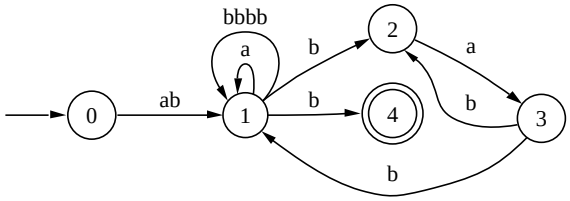
(i) DFA:



(ii) NFA:



(iii) NFA:

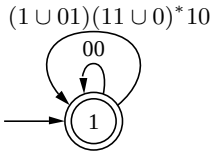


Solution 3.9.

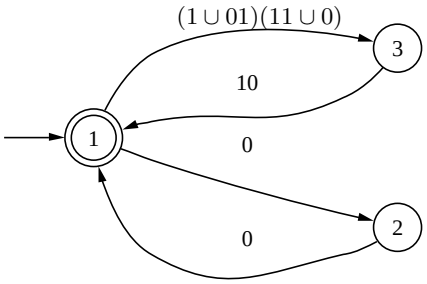
(o29)

Construct an NFA given a GFA.

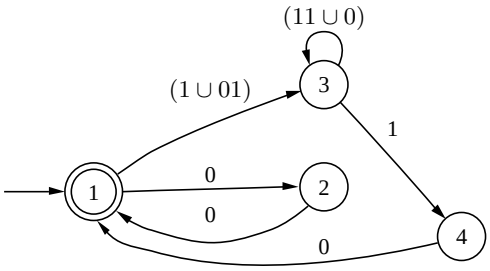
1.



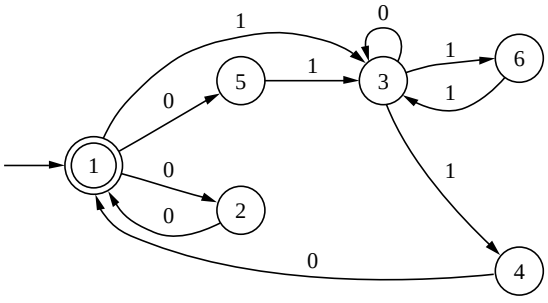
2.



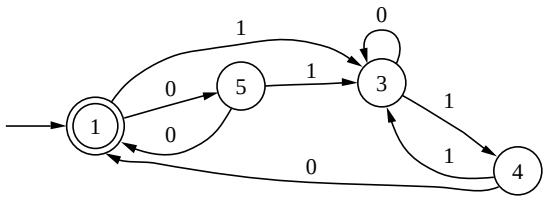
3.



4.



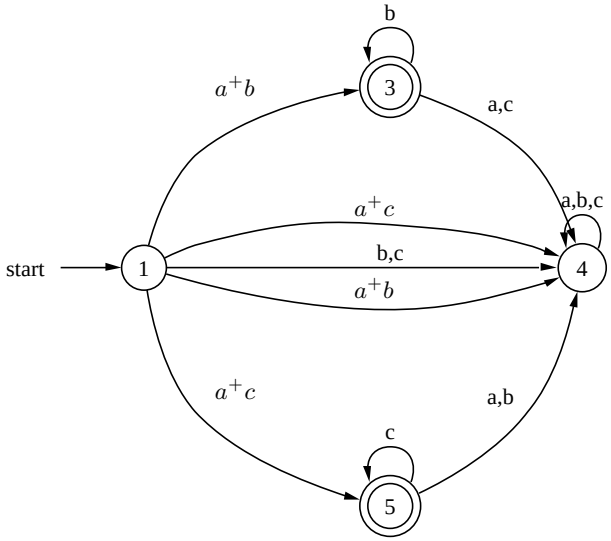
Convert the NFA to a DFA (join the obviously equivalent states $\{5, 2\}$ and $\{4, 6\}$).



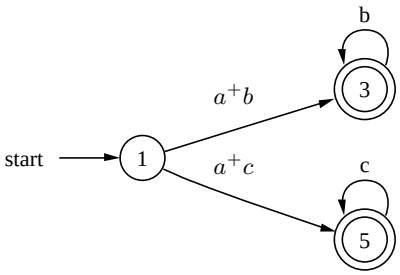
Solution 3.10. (o19)

We construct a GFA from the finite automation.

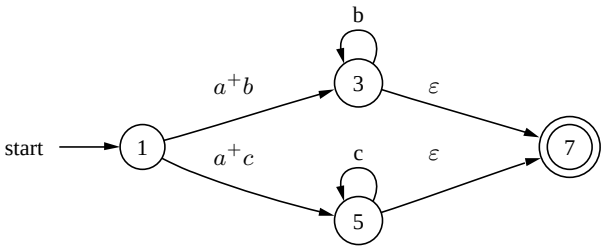
1. Eliminate states 2 and 6, new transitions $(1, 3), (1, 4), (1, 5), (1, 4)$.



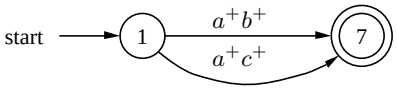
2. Eliminate state 4. Since there are no paths from state 4 (dead state) we can remove it entirely.



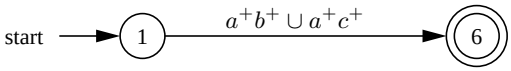
3. Create a new accepting state, number 7, and make new epsilon-transitions from the previous accepting states to 7. We do this to be able to eliminate state 3 and 5.



4. Eliminera tillstånd 3 och 5.



5. Join the two transitions from 1 to 7.



Thus, the automation accepts the language $a^+b^+ \cup a^+c^+$, which is the same as $\{a^ib^j, a^ic^j \mid i, j \geq 1\}$, which concludes the proof. □

Solution 3.11. (o35)

(i) b^*a

(ii) $b^*a(a \cup b)^*$

(iii) $(a \cup baa)^*b(a \cup \varepsilon)$

Solution 3.12. (o27)

Construct GFA's to create regular expressions, see Salling page 48 and forward.

(i) $\big(b^+a(b \cup ab^*a)\big)^*\big(ab^* \cup b^+a(\varepsilon \cup ab^*)\big)$

(ii) $\big(ab \cup aa(aa)^*ab\big)^*\big(a \cup aa(aa)^*(\varepsilon \cup a)\big)$

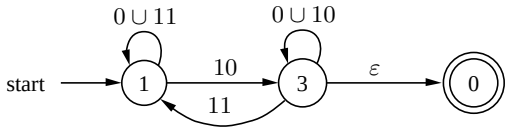
(iii) $a(aa \cup ab)^*$

(iv) $b^*a(b^+ab^*a)^*$

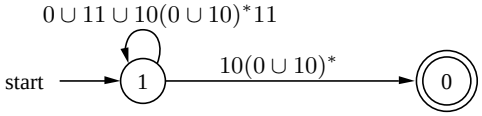
Solution 3.13. (o30)

We start by constructing the corresponding GFA.

1. New final state, remove state 2.



2. Remove state 3.

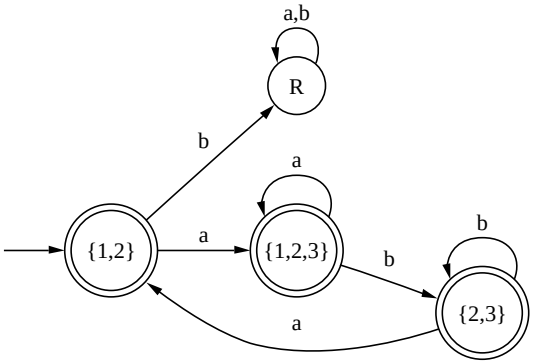


We now obtain the regular expression $\big(0 \cup 11 \cup 10(0 \cup 10)^*11\big)^*10(0 \cup 10)^*$.

Solution 3.14. (○38)

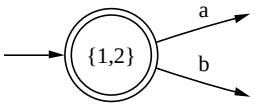
We solve the assignments by the method from Salling, page 41.

(i)

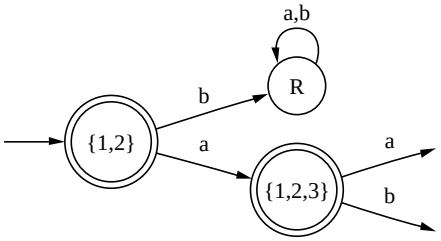


Step by step:

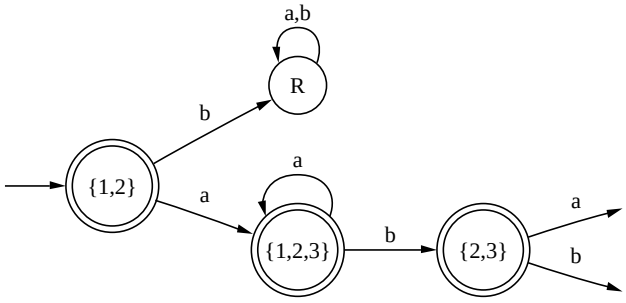
1. Initial state: $\{1, 2\}$



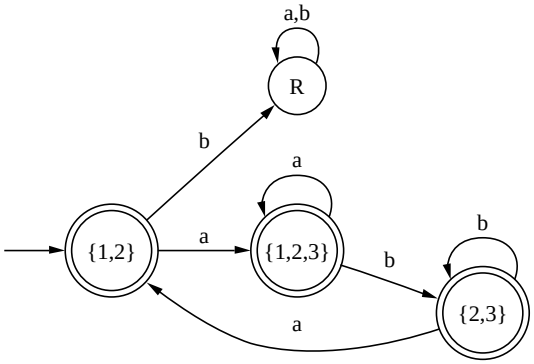
2. $\{1, 2\} \xrightarrow{a} \{1, 2, 3\}$
   $\{1, 2\} \xrightarrow{b} \{\}$(dead state)



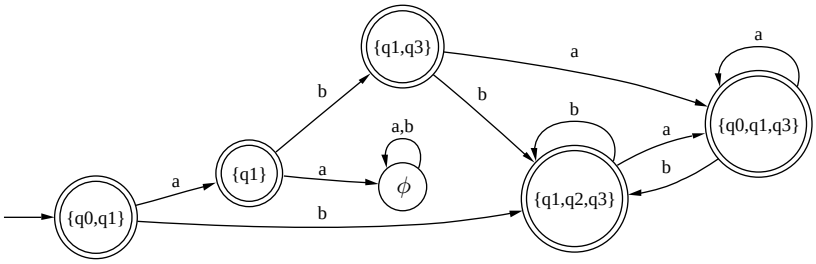
3. $\{1, 2, 3\} \xrightarrow{a} \{1, 2, 3\}$
   $\{1, 2, 3\} \xrightarrow{b} \{2, 3\}$



4. $\{2,3\} \xrightarrow{a} \{1,2\}$
   $\{2,3\} \xrightarrow{b} \{2,3\}$



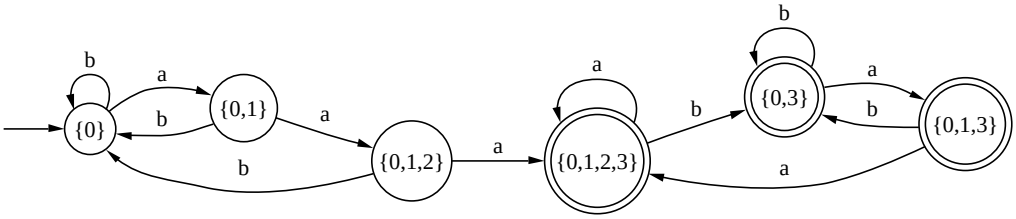
(ii)



1. Initial state: $\{q0, q1\}$
2. $\{q0, q1\} \xrightarrow{a} \{q1\}$
3. $\{q0, q1\} \xrightarrow{b} \{q1, q2, q3\}$
4. $\{q1\} \xrightarrow{a} \{\}$ (dead)
5. $\{q1\} \xrightarrow{b} \{q1, q3\}$
6. $\{q1, q3\} \xrightarrow{a} \{q0, q1, q3\}$
7. $\{q1, q3\} \xrightarrow{b} \{q1, q2, q3\}$
8. $\{q0, q1, q3\} \xrightarrow{a} \{q1, q2, q3\}$
9. $\{q0, q1, q3\} \xrightarrow{b} \{q1, q2, q3\}$
10. $\{q1, q2, q3\} \xrightarrow{a} \{q1, q2, q3\}$
11. $\{q1, q2, q3\} \xrightarrow{b} \{q1, q2, q3\}$

Solution 3.15. (○65)

We use the subset construction.



Step by step:

1. Initial state: $\{0\}$
2. $\{0\} \xrightarrow{a} \{0, 1\}$

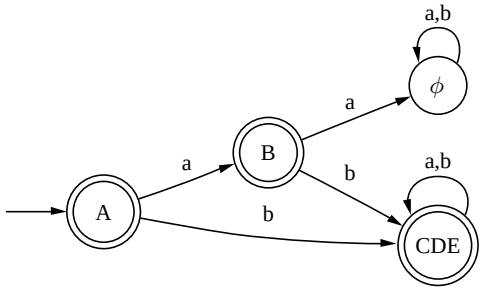
3. $\{0\} \xrightarrow{b} \{0\}$
4. $\{0,1\} \xrightarrow{a} \{0,1,2\}$
5. $\{0,1\} \xrightarrow{b} \{0\}$
6. $\{0,1,2\} \xrightarrow{a} \{0,1,2,3\}$
7. $\{0,1,2\} \xrightarrow{b} \{0\}$
8. $\{0,1,2,3\} \xrightarrow{a} \{0,1,2,3\}$
9. $\{0,1,2,3\} \xrightarrow{b} \{0,3\}$
10. $\{0,3\} \xrightarrow{a} \{0,1,3\}$
11. $\{0,3\} \xrightarrow{b} \{0,3\}$
12. $\{0,1,3\} \xrightarrow{a} \{0,1,2,3\}$
13. $\{0,1,3\} \xrightarrow{b} \{0,3\}$

Solution 3.16. (o39)

We show how to solve the problem both by using set partitioning and by using the double complement.

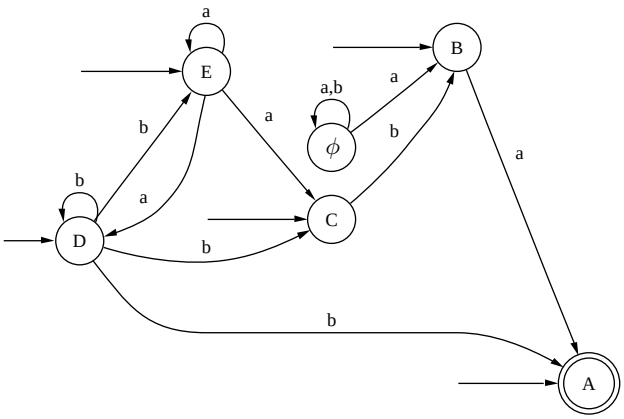
Set partitioning: We use the minimisation algorithm from Salling, page 60.

Level 0	$\{A,B,C,D,E,\emptyset\}$			
Level 1	$\{A,B,C,D,E\}$	$\{\emptyset\}$		The state $\emptyset$ is non-accepting and the states A,B,C,D,E are accepting.
Level 2	$\{A,C,D,E\}$	$\{B\}$	$\{\emptyset\}$	The state B transfers to $\emptyset$ by the string a, but the states A,C,D,E transfers to states in $\{C,D,E\}$.
Level 3	$\{C,D,E\}$	$\{A\}$	$\{B\}$	$\{\emptyset\}$	A transfers to $\emptyset$ by the string aa, but C,D,E transfers to states in $\{D,E\}$.
Level 4	$\{C,D,E\}$	$\{A\}$	$\{B\}$	$\{\emptyset\}$	No further states can be distinguished.



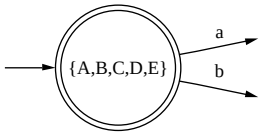
Double reversing: We use the double reversing algorithm from Salling, page 62.

- **First we reverse the automation and obtain an NFA M^{rev}.**

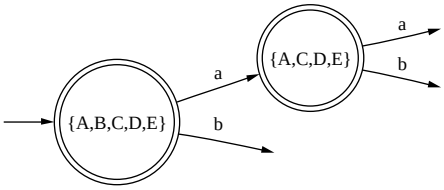


- **The next step is to convert M^{rev} to a DFA M' using subset construction.**

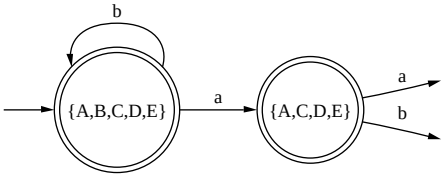
1. First we create a new state from the set of states in the NFA which are reachable by consuming ε from some initial state. Note that the new state is an accepting state, since the state A is accepting.



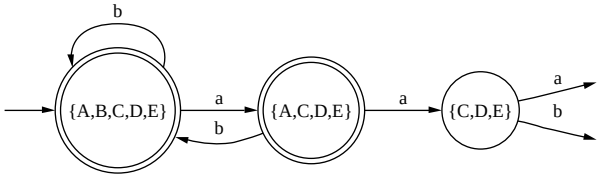
2. Then, we create a new state representing the reachable states from $\{A, B, C, D, E\}$ by consuming an a. The reachable states are $\{A, C, D, E\}$.



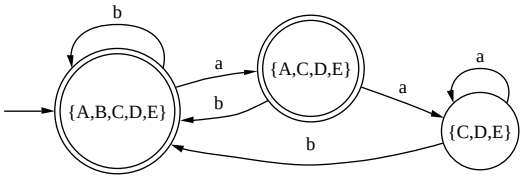
3. Now we find the states reachable by consuming a b from a state in $\{A, B, C, D, E\}$.



4. We carry on in the same manner, but from some state in $\{A, C, D, E\}$.

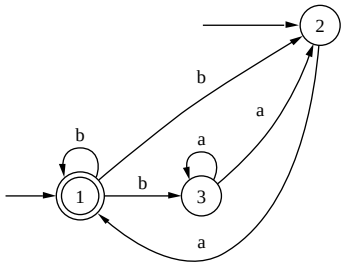


5. Over again, but from a state in $\{C, D, E\}$.

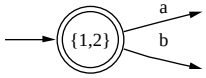


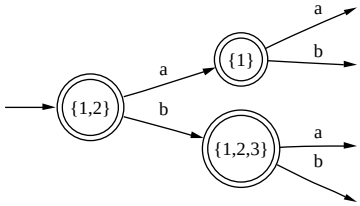
6. Now, we have obtained an equivalent DFA M' since no more transitions need to be examined.

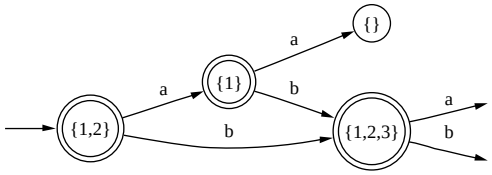
- **Now we reverse M' to obtain an NFA $(M')^{rev}$.**

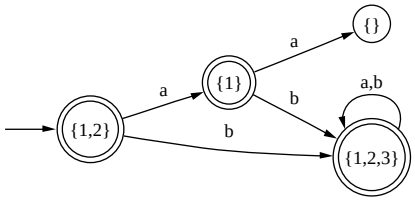


- **From $(M')^{rev}$ we create a DFA M'' as before.**

1. 

2. 

3. 

4. 

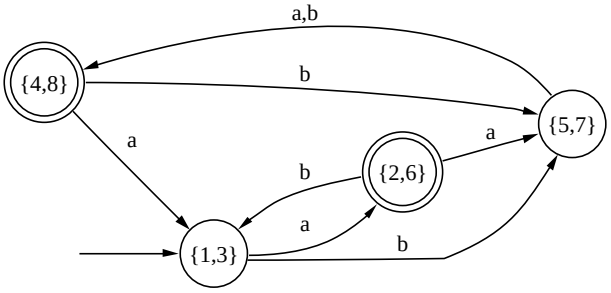
- **We are done! Note that this DFA is the same as the one we obtained earlier, except that the states have different names.**

Solution 3.17. (o80)

We solve the problem by subset construction, Salling page 60.

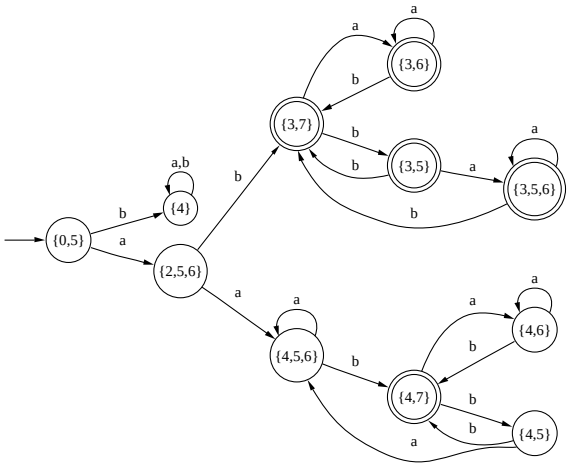
Level 0	$\{1,2,3,4,5,6,7,8\}$			$Start$
Level 1	$\{2,4,6,8\}$	$\{1,3,5,7\}$		The states 2,4,6 and 8 are accepting and the states 1,3,5 and 7 are not.
Level 2	$\{2,6,4,8\}$	$\{1,3\}$	$\{5,7\}$	The states 1 and 3 transfers to rejecting states by the string b, but the states 5 and 7 transfers into accepting states.
Level 3	$\{2,6\}$	$\{4,8\}$	$\{1,3\}$ $\{5,7\}$	The states 2 and 6 transfers into acceptingt states by the string ab, but the states 4 and 8 transfers into rejecting states by the string.
Level 4	$\{2,6\}$	$\{4,8\}$	$\{1,3\}$ $\{5,7\}$	No further distinguishing can be done.

The minimal DFA is shown below.



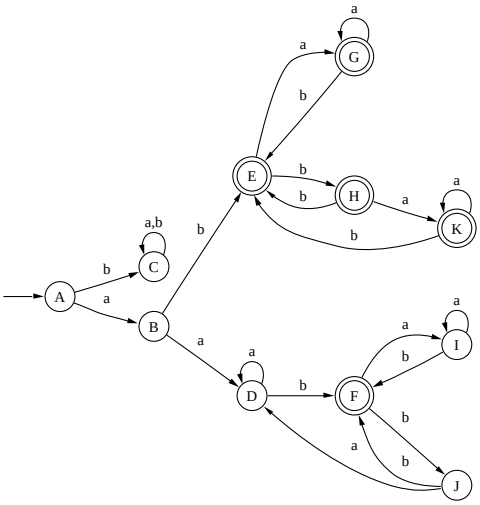
Solution 3.18. (o64)

Interpret the automata as one NFA and convert this to a DFA using subset construction.



Step by step:

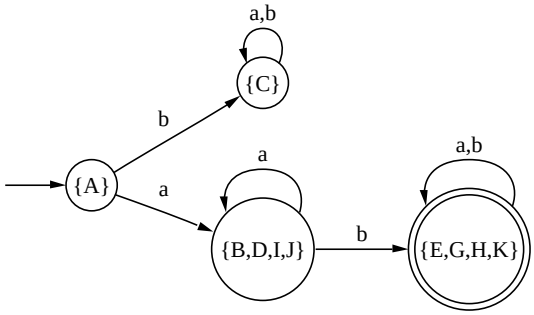
1. Initial state: $\{0,5\}$
2. $\{0,5\} \xrightarrow{a} \{2,5,6\}$
3. $\{0,5\} \xrightarrow{b} \{4\}$
4. $\{2,5,6\} \xrightarrow{a} \{4,5,6\}$
5. $\{2,5,6\} \xrightarrow{b} \{3,7\}$
6. $\{4\} \xrightarrow{a} \{4\}$
7. $\{4\} \xrightarrow{b} \{4\}$
8. $\{4,5,6\} \xrightarrow{a} \{4,5,6\}$
9. $\{4,5,6\} \xrightarrow{b} \{4,7\}$
10. $\{3,7\} \xrightarrow{a} \{3,6\}$
11. $\{3,7\} \xrightarrow{b} \{3,5\}$
12. $\{4,7\} \xrightarrow{a} \{4,6\}$
13. $\{4,7\} \xrightarrow{b} \{4,5\}$
14. $\{3,6\} \xrightarrow{a} \{3,6\}$
15. $\{3,6\} \xrightarrow{b} \{3,7\}$
16. $\{3,5\} \xrightarrow{a} \{3,5,6\}$
17. $\{3,5\} \xrightarrow{b} \{3,7\}$
18. $\{4,6\} \xrightarrow{a} \{4,6\}$
19. $\{4,6\} \xrightarrow{b} \{4,7\}$
20. $\{4,5\} \xrightarrow{a} \{4,5,6\}$
21. $\{4,5\} \xrightarrow{b} \{4,7\}$
22. $\{3,5,6\} \xrightarrow{a} \{3,5,6\}$
23. $\{3,5,6\} \xrightarrow{b} \{3,7\}$



We rename the states according to below.
Minimisation:

Level 0	$\{A,B,C,D,E,F,G,H,I,J,K\}$					
Level 1	$\{A,B,C,D,I,J\}$			$\{E,F,G,H,K\}$		Accepting or not
Level 2	$\{A,C\}$		$\{B,D,I,J\}$	$\{E,G,H,K\}$	$\{F\}$	foo
Level 3	$\{A\}$	$\{C\}$	$\{B,D,I,J\}$	$\{E,G,H,K\}$	$\{F\}$	bar
Level 4	$\{A\}$	$\{C\}$	$\{B,D,I,J\}$	$\{E,G,H,K\}$	$\{F\}$	No further distinguishing possible.

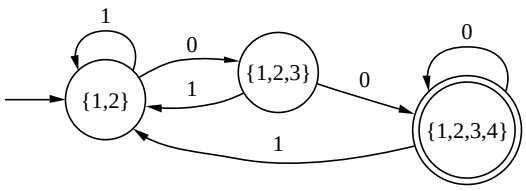
The new DFA:



☞ SOLUTION TEST ASSIGNMENT 2. (o28)

TBA.

☞ SOLUTION TEST ASSIGNMENT 3. (o11)



Step by step:

1. Initial state: $\{1, 2\}$
2. $\{1, 2\} \xrightarrow{0} \{1, 2, 3\}$
3. $\{1, 2\} \xrightarrow{1} \{1, 2\}$
4. $\{1, 2, 3\} \xrightarrow{0} \{1, 2, 3, 4\}$
5. $\{1, 2, 3\} \xrightarrow{1} \{1, 2\}$
6. $\{1, 2, 3, 4\} \xrightarrow{0} \{1, 2, 3, 4\}$
7. $\{1, 2, 3, 4\} \xrightarrow{1} \{1, 2\}$

☞ SOLUTION TEST ASSIGNMENT 4. (o72)

TBA.

4 Regular Grammar

Exercise 4.1. (o46)

Try to construct a regular grammar which generates the language $\{a^n b^n \mid n \in \mathbb{N}\}$. Explain why you don't succeed!

Exercise 4.2. (o48)

Create regular grammar for the following languages. Use S as the first non-terminal. When you have created the grammars, construct the corresponding NFA's.

(i) $\{w \mid w \in \{0,1\}^* \wedge w$ containts the substring 00 or 11$\}$

(ii) $1^* \cup 10^*1^*0$

Exercise 4.3. (o49)

Create NFA's for the following languages and construct the corresponding regular grammars using these. Use S as the first non-terminal.

(i) $(\varepsilon \cup 1)(1 \cup 0)^*01^*$

(ii) $\emptyset^*1^* \cup (0\emptyset)^*$

(iii) $\{w \mid w \in \{0,1\}^* \wedge w$ doesn't end in 01$\}$

Exercise 4.4. (o47)

Create a regular grammar for the language of string over $\{a, b\}$ which contains the substrings aa and bb.

Exercise 4.5. (o50)

Construct regular grammars for the following languages.

(i) $(aabb^*aab^*a)^* \cup (ba^*b)$

(ii) $\{a^n b^m \mid n + m = 2k + 1 \wedge n, m, k \in \mathbb{N}\}$

Exercise 4.6. (o77)

Construct a regular grammar which produces the language over the alphabet $\Sigma = \{a, b\}$ consisting of all strings with at most 3 a's. Show how the string $babbaab$ is produced.

Exercise 4.7. (o78)

Construct a regular grammar producing the following language over the alphabet $\Sigma = \{a, b\}$:

$$L = \{w \mid (|w|_a - |w|_b \equiv 1 \pmod 3)\}$$

Then show how the string $abaaaaba$ is produced.

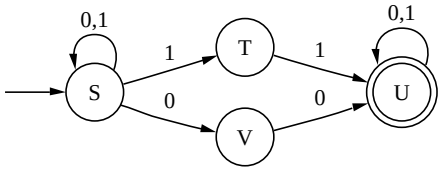
Solutions

Solution 4.1. (o46)

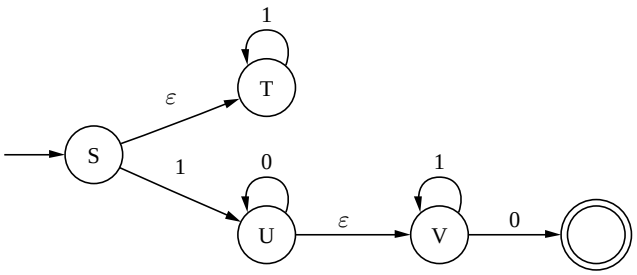
It is logical to start by $S \to a\ldots$ as production rule. From the beginning we have to be able to generate b's to obtain the same number of a's and b's. However, no other terminals may appear in a regular production rule. The context-sensitive grammar $S \to aSb \mid \varepsilon$, on the other hand, is generating the language.

Solution 4.2. (o48)

(i)
$$\begin{array}{lcl} S & \to & 0S \mid 1S \mid 1T \mid 0V \\ T & \to & 1U \\ V & \to & 0U \\ U & \to & 0U \mid 1U \mid \varepsilon \end{array}$$

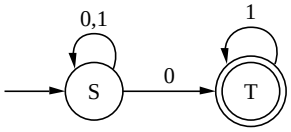


(ii)
$$\begin{array}{lcl} S & \to & T \mid 1U \\ T & \to & 1T \mid \varepsilon \\ U & \to & 0U \mid V \\ V & \to & 1V \mid 0 \end{array}$$



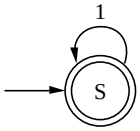
Solution 4.3. (o49)

(i) $(\varepsilon \cup 1)(1 \cup 0)^*01^* = (1 \cup 0)^*01^*$



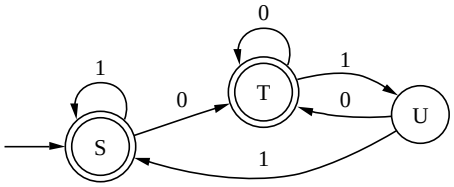
$$\begin{array}{lcl} S & \to & 0S \mid 1S \mid 0 \\ T & \to & 1T \mid \varepsilon \end{array}$$

(ii) $\emptyset^*1^* \cup (0\emptyset)^* = 1^*$



$$S \to 1S \mid \varepsilon$$

(iii)



$$\begin{array}{lcl} S & \to & 0T \mid 1S \mid \varepsilon \\ T & \to & 0T \mid 1U \mid \varepsilon \\ U & \to & 0T \mid 1S \end{array}$$

Solution 4.4. (o47)

$$\begin{aligned}
S &\rightarrow S_1 \mid S_2 \\
S_1 &\rightarrow aS_1 \mid bS_1 \mid aA \\
A &\rightarrow aB \\
B &\rightarrow aB \mid bB \mid bC \\
C &\rightarrow bD \\
D &\rightarrow aD \mid bD \mid \varepsilon \\
S_2 &\rightarrow aS_2 \mid bS_2 \mid bE \\
E &\rightarrow bF \\
F &\rightarrow aF \mid bF \mid aG \\
G &\rightarrow bH \\
H &\rightarrow aH \mid bH \mid \varepsilon
\end{aligned}$$

Solution 4.5. (o50)

(i)

$$\begin{aligned}
S &\rightarrow S_1 \mid S_2 \\
S_1 &\rightarrow aabA_1 \mid \varepsilon \\
A_1 &\rightarrow bA_1 \mid B_1 \\
B_1 &\rightarrow aaC_1 \\
C_1 &\rightarrow bC_1 \mid D_1 \\
D_1 &\rightarrow aS_1 \\
S_2 &\rightarrow bA_2 \\
A_2 &\rightarrow aA_2 \mid B_2 \\
B_2 &\rightarrow b
\end{aligned}$$

(ii) Construct strings with odd length consisting of some number of a's followed by some number of b's.

$$\begin{aligned}
S &\rightarrow aA_{\text{odd}} \mid bB_{\text{odd}} \\
A_{\text{odd}} &\rightarrow aA_{\text{even}} \mid bB_{\text{even}} \mid \varepsilon \\
A_{\text{even}} &\rightarrow aA_{\text{odd}} \mid bB_{\text{odd}} \\
B_{\text{odd}} &\rightarrow bB_{\text{even}} \mid \varepsilon \\
B_{\text{even}} &\rightarrow bB_{\text{odd}}
\end{aligned}$$

Solution 4.6. (o77)

$$\begin{aligned}
S &\rightarrow bS \mid aS \mid \varepsilon \\
A &\rightarrow bA \mid aB \mid \varepsilon \\
B &\rightarrow bB \mid aC \mid \varepsilon \\
C &\rightarrow bC \mid \varepsilon
\end{aligned}$$

Production of the string $babbaab$:

$$S \Rightarrow bS \Rightarrow baS \Rightarrow babA \Rightarrow babbA \Rightarrow babbaB \Rightarrow babbaaC \Rightarrow babbaabC \Rightarrow babbaab$$

Solution 4.7. (○78)

$$\begin{array}{lcl} S & \rightarrow & aA \mid bB \\ A & \rightarrow & aB \mid bS \mid \varepsilon \\ B & \rightarrow & aS \mid bA \end{array}$$

The string $abaaaaba$ is produced as follows.

$$S \Rightarrow aA \Rightarrow abS \Rightarrow abaA \Rightarrow abaaB \Rightarrow abaaaS \Rightarrow abaaaaA \Rightarrow$$

$$abaaaabS \Rightarrow abaaaabaA \Rightarrow abaaaaba$$

5 Non-Regularity

Exercise 5.1. (o79)

Show that if the languages L_1 and L_2 are regular, then the following language are regular. You can use any proof technique you like, and use any proof from the course book and/or the lecture material.

(i) $L_1 L_2$

(ii) L_1^*

(iii) $L_1 \cup L_2$

(iv) $\overline{L_1}$, dvs. $\Sigma^* - L_1$

(v) $L_1 \cap L_2$

(vi) $L_1 - L_2$

(vii) $\{w \mid w \in L_1$ if and only if $w \notin L_2\}$

(viii) $prefix(L_1)$ (see Notation)

(ix) $suffix(L_1)$ (see Notation)

(x) $substring(L_1)$ where $substring(L) = \{w \mid \exists x, y$ such that $xwy \in L\}$

Exercise 5.2. (o68)

Which of the following languages are regular?

(i) $\{a^n b^m \mid n > m \vee n < m\}$

(ii) $\{a^n b^m \mid n \geq m \vee n \leq m\}$

(iii) $\{a^n b^m \mid n > m \wedge n < m\}$

(iv) $\{a^n b^m \mid n \geq m \wedge n \leq m\}$

Exercise 5.3. (o34)

Prove that if L is regular, then L^{rev} is regular.

Exercise 5.4. (o70)

What can be said about the language $L = \{w \in \{a, b\}^* \mid abw = wba\}$? is it the empty string? is it regular?

Exercise 5.5. (o31)

Does it hold that a language cannot be regular if it is the union of languages which are non-regular? Does it hold for the intersection? For concatenation? If it is true, prove it. If not, give a counter example.

Exercise 5.6. (o82)

Determine if the following languages are regular or non-regular. Prove that your answers are correct. You may use properties of regular languages, closure properties of regular languages, things you know about DFA's, NFA's or regular expressions and grammars.

(i) $\{a^n b^{2n}\}$

(ii) $\{w \mid w \in \{0,1\}^* \wedge |w|_0 \neq |w|_1\}$

(iii) $\{w \mid w \in \{a,b\}^* \wedge |w|_a = 2|w|_b\}$

(iv) $\{w^{rev} \mid w \in L(a^n b^m)\}$

(v) $\{xwx^{rev}y \mid x,w,y \in \{a,b\}^+\}$

(vi) $\{a^n b^{n+m} c^m \mid m \leq 2\}$

Exercise 5.7. (o32)

Prove that the following languages are non-regular:

(i) $\{0 \cdot 1^n 0^{m+n} \mid m,n \geq 1\}$

(ii) $\{ww \mid w \in (0 \cup 1)^+\}$

(iii) $\{(01)^n (10)^n \mid n \geq 0\}$

(iv) The set of strings over $\{0,1\}$ with the equally many zeroes as ones.

(v) $\{w \in (0 \cup 1)^* \mid w = w^{rev}\}$

(vi) $\{xx^{rev}w \mid x,w \in (0 \cup 1)^+\}$

(vii) $\{0^n \mid n \text{ is prime}\}$

(viii) $\{0^n \mid n \text{ is composite}\}$

(ix) $\{0^m 1^n \mid m \neq n\}$

(x) $\{0^m 1^n \mid gcd(m,n) = 1\}$, where the function $gcd(x,y)$ is the greatest common divisor.

Exercise 5.8. (o81)

Prove that the language $L = \{a^n b^n\}, n \in \mathbb{N}$ is non-regular using the pumping lemma.

Exercise 5.9. (o62)

Prove that the language $L = \{w \in \{0,1\}^* \mid \exists u : www = uu\}$ is non-regular.

Exercise 5.10. (o67)

Let L be a regular language. What can be said about the following language?

$$L_1 = \{w \mid ww^{rev}ww^{rev} \in L\}$$

Exercise 5.11. (o73)

x is a *root* to w if $\exists n \geq 0$ such that $w = x^n$. We define the root to a language L as $\mathrm{rot}(L) = \{x \mid \exists n \geq 0 : x^n \in L\}$. Prove that for each language $L \subseteq \Sigma^*$ it holds that $\varepsilon \in L \Leftrightarrow \Sigma^* = \mathrm{root}(L)$.

☞ TEST ASSIGNMENT 5. (LESSON) (o14)

Determine if the following languages are regular or not.

(i) $L_a = \{a^i b^j c^k \mid i + j = k\}$

(ii) $L_b = \{a^i b^j c^k \mid 1 \leq i \leq j \leq k \leq \min(81, j^2)\}$

Solutions

Solution 5.1. (o79)

(i) Regular languages are by definition closed under concatenation.

(ii) Regular languages are by definition closed under the Kleene star operation.

(iii) Regular languages are by definition closed under union.

(iv) If L_1 is regular, it is possible to construct a DFA for the language. Assume that $M_1 = (Q, \Sigma, \delta, s, F)$ is such a DFA. Then it is possible to construct a new DFA $M_1' = (Q, \Sigma, \delta, s, Q - F)$ which rejects a string w if and only if M_1 would accept w. $\therefore L(M_1') = \Sigma^* - L(M_1) = \Sigma^* - L_1 = \overline{L_1}$. Since it exists a DFA accepting the language $\overline{L_1}$, then $\overline{L_1}$ is regular if L_1 is regular. $\therefore$ Regular languages are closed under complement. □

(v) DeMorgans law implies that $L_1 \cap L_2 = \overline{\left(\overline{L_1} \cup \overline{L_2}\right)}$. Since regular languages are closed under complement and union, they are also closed under intersection.

(vi) Note that $L_1 - L_2 = L_1 \cap \overline{L_2}$. Since regular languages are closed under intersection and complement, they are also close under set difference.

(vii) Note that $\{w \mid w \in L_1$ om och endast om $w \notin L_2\} = (L_1 - L_2) \cup (L_2 - L_1)$. Since regular languages are closed under set difference and union, this set is regular.

(viii) Assume that M_1 is a NFA accepting L_1. Construct a new NFA M_1' equivalent to M_1 except that we add a new state q_f, which is the only final state in M_1'. We also add ε-transitions to q_f from all states from with a final state is reachable in M_1. If $wz \in L_1$, then w is consumed by the regular transitions in M_1, followed by one of the ε-transitions to q_f to simulate the consumption of z, without actually consuming something. Thus, $L(M_1') = \mathit{prefix}(L(M_1)) = \mathit{prefix}(L_1)$, that is, if L_1 is regular, then $\texttt{prefix}(L_1)$ is also regular.

(ix) Assume that M_1 is a NFA accepting L_1. Construct a new NFA M_1' equivalent to M_1 except that we add a new initial state q_0. In addition, we add ε-transitions from q_0 to all states where an initial state is reachable. If $xw \in L_1$, then the consumption of x is simulated by a transition from the first set of ε-transitions and w is consumed by the regular transitions in M_1 . Thus, $L(M_1') = \mathit{suffix}(L(M_1)) = \mathit{suffix}(L_1)$, and so it holds that if L_1 is regular, then $\mathit{suffix}(L_1)$ is also regular.

(x) Assume that M_1 is an NFA accepting L_1. Construct a new NFA M_1' equivalent to M_1 except that we add a new initial state q_0 and a new final state q_f, which is the only final state in M_1'. In addition, we add ε-transitions q_0 to all states from which an initial state can be reached, and ε-transitions q_f from all states from which a final state can be reached in M_1. If $xwy \in L_1$, the consumption of x is simulated by a transition from the first set of ε-transitions, w is then consumed by the regular transitions of M_1 and the consumption of y is simulated bya transition from the second set of ε-transitions. Thus, $L(M_1') = \mathit{delstrang}(L(M_1)) = \mathit{delstrang}(L_1)$, and hence it holds that if L_1 is regular, then $\mathit{delstrang}(L_1)$ is also regular.

Solution 5.2. (o68)

- Not regular. It is the same language as $\{a^n b^m \mid n \neq m\}$ which is the complement to $\{a^n b^m \mid n = m\}$ which is not regular (which you should have discovered earlier). Since regularity is closed under complementation, the considered language can not be regular (since it would imply that $\{a^n b^m \mid n = m\}$ is regular).
- Regular. The language can be described by the regular expression a^*b^*.
- Regular. This language contains no language, and is thus regular.

Solution 5.3. (o34)

Since L is regular, there exists a DFA accepting this language. If we reverse all transitions in this automation, and turn the initial state into an accepting state and all accepting states to initial states, the result is that we run the original automation "backwards", and the new automation will accept the language L^{rev}. Since there exists an finite automation accepting L^{rev}, the language is regular.

Solution 5.4. (o70)

It is not the empty set since, for instance, $w = a$ and $w = aba$ satisfies $abw = wba$. If we look a bit closer at this condition we notice that w has to start with ab since this is the prefix on the left hand side. In the same way, the strings has to end in ba since that is the postfix on the right hand side. However, we can make an exception for the string a which may act both as initial and final a and "borrows", so to speak, its b's from the prefix and postfix strings in the condition. Thus, the string a satisfies the condition $aba = aba$. We now try to find some larger examples. The strings are supposed to start with ab, so we had to add a b to our minimal string a. In addition, we had the requirement that the strings should end in ba, but we can let the final b be the same as the initial one to obtain the string aba. If we keep on adding characters to this string, the requirement that they should end in ba yields the following strings $ababa, abababa, ababababa, \ldots$. As can be seen, there is a regularity in the strings which can be summarised as: the strings begins with an a followed by zero or more ba's. This can be formulated as the regular expression $a(ba)^*$. Since there exists a regular expression describing the language, it is regular.

Solution 5.5. (o31)

Languages formed by the union of two non-regular languages may be regular. For instance, this is the case with the language constructed by the union of the two non-regular languages $\{a^n b^m \mid n = m\}$ and $\{a^n b^m \mid n \neq m\}$, which is described by the regular expression a^*b^*.

The intersection of two non-regular languages may also result in a regular language, for instance the non-regular languages $a^n b^n$ and $b^n a^n$, which intersection only contains the empty string, which is regular.

Concatenation of two non-regular languages will however always result in a non-regular language. If we assume the contrary, that is, that it is possible to create a regular language by concatenation of two non-regular languages, there would be a finite automation accepting this language. Concatenation of languages corresponds to concatenation of two automata accepting respective language. Thus, it would be possible to split this finite automation into two separate automata accepting each of the original languages, which is a contradiction since these languages are non-regular which means there cannot exist finite automata accepting them.

Solution 5.6. (o82)

- Non-regular. Assume that $L = \{a^n b^{2n}\}$ is a regular language. Then there exists a DFA M accepting L. Create a new DFA M' equivalent to M except that all pairs of transitions $\delta(q1, b) = q2$ and $\delta(q2, b) = q3$ are changed to a single transition $\delta(q1, b) = q3$ (two transitions in M consuming two b's are replaced by a transition consuming one b). M' consumes half as many b's as M, $L(M') = \{a^n b^n\}$ which has to be regular, since there is a DFA accepting this language. This is a contradiction since we know that $\{a^n b^n\}$ is non-regular. Thus, L is also non-regular.

- Non-regular. Assume that $L = \{w \mid w \in \{0,1\}^* \wedge |w|_0 \neq |w|_1\}$ is regular. Since regular languages are closed under complementation and intersection, $\neg(L) \cap L(0^*1^)$ has to be regular. However, $\neg(L) \cap L(0^*1^*) = \{0^n 1^n\}$, which is not regular, and thus neither is L.

- Non-regular. Assume that $L = \{w \mid w \in \{a,b\}^* \wedge |w|_a = 2|w|_b\}$ is regular. We know from the closure properties of regular languages that $L \cap L(a^*b^*)$ is regular. However, $L \cap L(a^*b^*) = \{a^n b^{2n}\}$, which is not regular. Hence, L is cannot be regular.

- Regular. The regular expression b^*a^* describes the language.
- Regular. The regular expression $a(a \cup b)^+a(a \cup b)^+ \cup b(a \cup b)^+b(a \cup b)^+$ describes this language. Note that w can have an arbitrary length, so it may consume the trail of x and the beginning of x^{rev}. The only requirement is that the first character in x (which must contain at least one character) is present in the rest of the string.
- Non-regular. This language can also be expressed as $L = \{a^n b^n\} \cup \{a^n b^{n+1} c\} \cup \{a^n b^{n+2} c^2\}$. Assume that L is regular. Then, $L \cap \{a^*b^*\}$ has to be regular. However, $L \cap \{a^*b^*\} = \{a^n b^n\}$, which is non-regular, thus L cannot be regular.

Solution 5.7. (o32)

- $\{01^n 0^{n+m} \mid m, n \geq 1\} = \{0\}\{1^n 0^n\}\{0^m\}$. Since we know that $\{1^n 0^n\}$ and $\{0^m\}$ are non-regular and that concatenation of non-regular languages are non-regular, the language cannot be regular.
- The language is non-regular. Since each pair of the infinitely many strings over $(0 \cup 1)^+$ are distinguished by the language, that is, each pair $x, y \in (0 \cup 1)^+$) where $x \neq y$ is distinguished when xx belongs to the language and yx doesn't.
- The fact that this language is non-regular might be simpler to understand if we let $a = 01$ and $b = 10$ resulting in the language $\{a^n b^n \mid n \geq 0\}$ which is known to be non-regular.
- Call this language L and assume that it is regular. Then $L \cap (0^*1^*)$ is also regular since regular languages are closed under intersection. However, this is a contradiction since $L \cap L(0^*1^*) = \{0^n 1^n\}$, which is non-regular. Thus, L cannot be regular.
- This language distinguishes an infinite number of strings and it therefore non-regular. Every pair of the infinitely many strings $x, y \in (0 \cup 1)^+$ where $x \neq y$ are distinguished when xx^{rev} belongs to the language but not yx^{rev}.
- The language has to distinguish infinitely many strings. The proof is similar to the solution above. For each of the infinitely many strings $x, y \in (0 \cup 1)^+$ where $x \neq y$, there exists a string $z = x^{rev}$ such that xzw belongs to the language, but not yzw. Hence infinitely many strings are distinguished, and the language is not regular.
- Also here it is useful to use argumentation about distinguishing strings. However, we need to make slightly stricter requirements on the strings since it otherwise would not be obvious that the language distinguishes between them. Take two strings $x = 0^n$ and $y = 0^{n+1}$. The number of such strings are infinite. Since there exists an infinite number of prime numbers, there exists a string $z = 0^m$ such that $n + m$ is prime, but not $(n + 1) + m$ (except when $n = m = 1$ where both 2 and 3 are prime). Then xz belongs to the language, but not yz. The language then distinguishes an infinite number of strings, thus making it non-regular.
- Composite numbers are numbers which are not prime. Thus, this language is the complementary language to the language or prime strings in the previous exercise, which was non-regular. Considering this and the closure properties of complementation, we can derive that this language cannot be regular.
- This language is the complement language to $\{0^n 1^n\}$ which is non-regular. By the closure properties for complementation we know that also this language is non-regular.

Solution 5.8. (o81)

We start by taking a look at the pumping lemma.

The pumping lemma states properties that must hold for all strings in a regular language (with an infinite number of strings). The lemma states that all strings in such a language, which has a

length exceeding a certain number, contains a non-empty substring which may both be "pumped up" (repeated) an arbitrary number of times or "pumped out" (removed) such that the resulting string still is in the language. In addition it holds that such a substring exists in each substring of a string in the language, if the substring has a length equal to or exceeding the number stated above.

In other words, it holds that all strings $w \in L$ such that $|w| \geq N$ can be re-written as $w = xyz$ where $|y| \neq 0$, and for all $i \in \mathbb{N}$ it holds that $xy^iz \in L$. In addition, it holds that all substrings of w where $|w| \geq N$ has such a "pumping block" y.

To prove that a language is non-regular using the pumping lemma, it is easiest to try to construct a counter example. Thus, we start by assuming that the language $L = \{a^nb^n\}$ is regular, and try to achieve a contradiction. We will do this by observing an individual string in L and show that it is not possible to pump it using the pumping lemma.

Thus, according to the pumping lemma it holds that each string in L having a length greater tha or equal to an unknown number N should have a "pumping block" y which can be pumped. Since we have no idea what N would be, we have to be careful and select a string which is guaranteed to have a length which is equal to or greater than N.

The only way we can do this is to choose a string which is parameterised in N. For instance, we can consider the string $w = a^Nb^N$, for which it naturally holds that $|w| \geq N$.

The next step is to find out exact where in the string $w = xyz$ the pumping block y is. Here we could use the cact that a pumping block has to exist in all substrings of w where $|w| \geq N$.

Consider the substring a^N, for which it holds that $|a^N| \geq N$. Thus, a^N must contain a y such that $w = xy^iz, i \in \mathbb{N}$. So we have $w = a^{N-k}a^kb^N$ where $x = a^{N-k}$, $y = a^k$ and $z = b^N$. We attempt to pump out the pumping block y once, and obtain the string $a = a^{N-k}b^N$ where $k > 0$. This string is supposed to belong to the language L according to the pumping lemma (under the assumption that L is regular). But since we the number of a's and the number of b's are not the same, the pumped string *cannot* be in the language! We have found a contradiction, and have to reject our hypothesis that L would be regular. Hence, L is non-regular. □

Solution 5.9. (o62)

We use the pumping lemma also in this proof. The string $w = 10^N110^n1$ belongs to the language, since $www = uu$ with $u = 10^N110^N110^N1$. If the starting string of zeroes is pumped, which according to the pumping lemma should be possible since the length of this string is greater than or equal to N, the string is no longer in the language. That is, if $w = 10^{2N}110^N1$ then $www \neq uu$. Thus, the language is non-regular.

Solution 5.10. (o67)

The strings in the language consist of a concatenated pair of the substring ww^{rev}. Thus, the strings are the same backwards and forwards, i.e., they are palindromes. Palindrome languages are not regular, and as we saw earlier, it is not possible to obtain a regular language by concatenating two non-regular languages.

Solution 5.11. (o73)

We have that $root(L) = \{x \mid \exists n \geq 0 : x^n \in L\}$. Assume that $\varepsilon \in L$ and let $n = 0$. This results in $root(L) = \{x \mid x^0 \in L\}$ and since we no longer have any requirements on which strings x which should belong to $root(L)$ we have that $root(L) = \Sigma^*$. Thus, we have shown that $\varepsilon \in L \Rightarrow \Sigma^* = root(L)$.

Now assume that $\Sigma^* = root(L)$. We want to show that this implies $\varepsilon \in L$. Since $\varepsilon \in \Sigma^*$, it has to hold that $\exists n \geq 0 : \varepsilon^n \in L$. No matter how many times we repeat ε, the result is again ε, i.e., $\forall n \geq 0 : \varepsilon^n = \varepsilon$. Thus, it must hold that $\varepsilon \in L$. Consequently, we have shown that $\varepsilon \in L \Leftarrow \Sigma^* = root(L)$ and this together with the paragraph above yields that $\varepsilon \in L \Leftrightarrow \Sigma^* = root(L)$.

☞ SOLUTION TEST ASSIGNMENT 5. (o14)

- L_a is non-regular. Assume that L_a is regular for contradiction. Then there should be a number N such that if $w \in L_a$ and $|w| \geq N$, then w contains a non-empty substring y which can be pumped such that w stays in the language. Such a string exists, as a matter of fact, in every substring of w with a length of at least N characters. Consider the string $a^N b^N c^{2N}$ which belongs to the language since $N + N = 2N$. Let the beginning sequence of a's be the pumping string y, which satisfies the condition in the pumping lemma since $|y| \geq N$. If we pump up y once, the string no longer belongs to the language, since $2N + N \neq 2N$. Thus, we have achieved a contradiction, and L_a cannot be regular.

- L_b is regular. Each string in L_b may consist of maximum 81 of each of the characters a, b and c. This makes the number of strings in L_b finite, thus regular.