Universität Zürich
Institut für Informatik

Autumn 2020
Sven Seuken
Dennis Komm

Student Name:
Student Number:

Foundations of Computing II
Assignment 1 – Solutions

Formal Languages, Automata, Regular Expressions

Distributed: 21.09.2020 – Due Date: 04.10.2020

Upload your solutions to the OLAT system.

1.1 Alphabets, Words, Languages

a) Let $X = \{1321, 2222, 31\}$, $Y = \{\varepsilon, 11, 21\}$, and $Z = \{\varepsilon, u, bddd\}$ be languages over the alphabet $\{1, 2, 3, u, b, d\}$; let $\circ$ denote the concatenation operator.

(i) Give the set of strings in X^* that are of length 4.

(ii) Give the set of strings in $X \circ Y$ that are of length 6.

(iii) Give the set of strings in $(Y \cup Z) \circ X$ that are of length 5 or less.

(i) $\{1321, 2222, 3131\}$.

(ii) $\{132111, 132121, 222211, 222221\}$.

(iii) $\{1321, 2222, 31, 1131, 2131, u1321, u2222, u31\}$.

b) We consider two languages $\{1\}$ and $\{2\}$ that contain only one word each. You are only asked to explain your arguments in words; no formal arguments are required.

(i) Explain why $(\{1\}^*\{2\}^*)^* = (\{1, 2\}^*)^2$.

(ii) Explain why $(\{1\}^*\{2\}^*)^* \neq (\{1, 2\}^2)^*$.

(i) First, we note that $\{1, 2\}^*$ contains the empty word ε, and since, for every string x, $\varepsilon x = x$, $(\{1, 2\}^*)^2$ is nothing else than $\{1, 2\}^*$. This means that words of all possible lengths are contained in this language, and not just the ones of even length (as a first look might suggest). Then again, $(\{1\}^*\{2\}^*)^* = \{1, 2\}^*$ is certainly true, and thus both languages are the same.

(ii) Here, it is different. The language $\{1, 2\}^2$ only contains words of length 2 over $\{1, 2\}$, and therefore $(\{1, 2\}^2)^*$ contains all words over $\{1, 2\}$ of an even length. $(\{1\}^*\{2\}^*)^*$, on the other hand, also contains words of odd length, for instance, 1. Therefore, these two languages cannot be equal.

1.2 Finite Automata

If you want to create graphs, TIKZ is a nice tool to generate diagrams from code.

a) Draw (either by hand or by using a drawing tool) a finite automaton (DFA) for each of the following languages.

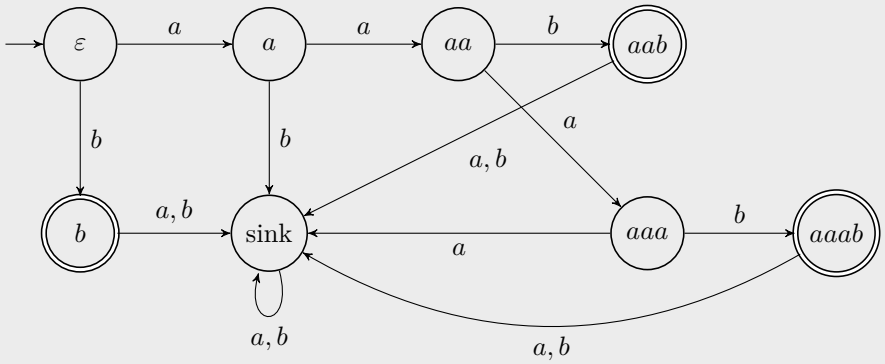
(i) The language $L_1 = \{aab, aaab, b\}$ over the alphabet $\{a, b\}$.

(ii) The language L_2 over the alphabet $\{a, b, c\}$ consisting of all words that start with aba and contain at least one c.

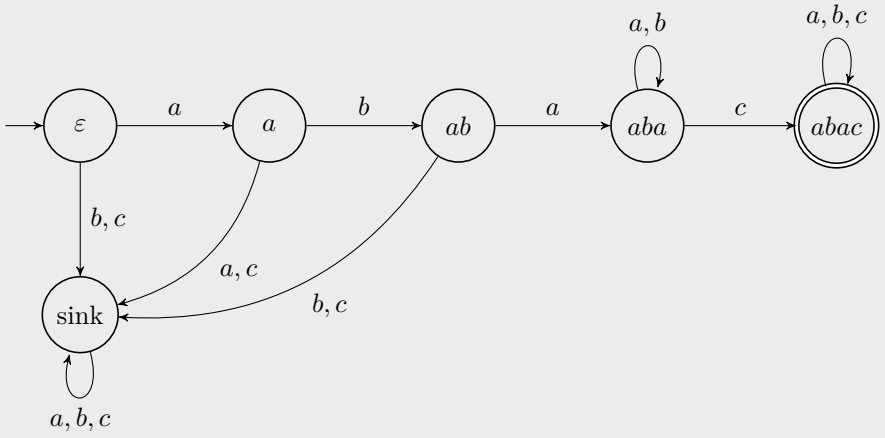
(iii) The language L_3 over the alphabet $\{0, 1\}$ that consists of the words that are the binary representation of even numbers. All representations (except 0) should start with a 1; for instance, the word 100 is in L_3, but 11, 0100, and 101 are not in L_3.

We label the states in a way that intuitively describes what they represent.

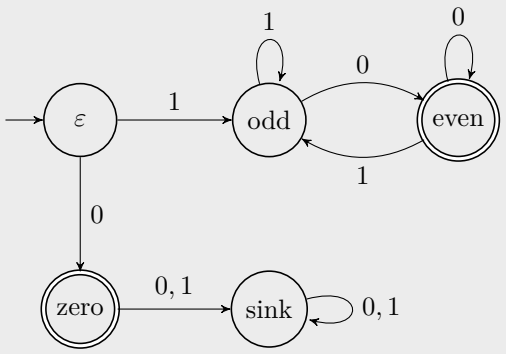
(i) DFA for L_1:



(ii) DFA for L_2:



(iii) DFA for L_3:



b) It is important to get the quantifiers straight in this context. All the following languages are over the alphabet $\{0, 1\}$.

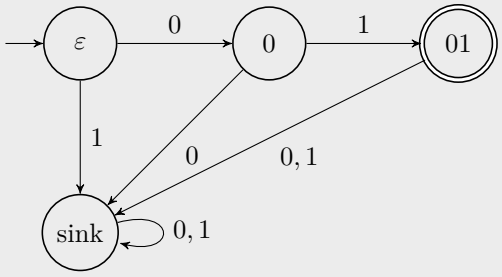
(i) Draw a DFA for the language $L_1 = \{01\}$.

(ii) Draw a DFA for the language $L_2 = \{0011\}$.

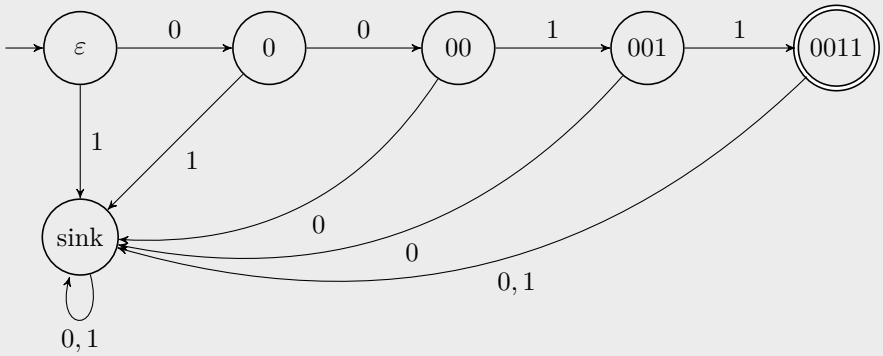
(iii) This can be generalized for arbitrary natural numbers. For a given $k \in \mathbb{N}$, sketch how an automaton for the language $L_k = \{0^k 1^k\}$ would look like.

(iv) However, explain on an intuitive level, in two or three sentences, where the problem lies, if one would want to create a DFA for the language $L = \{0^k 1^k \mid k \in \mathbb{N}\}$. Note that L contains all words $\varepsilon, 01, 0011, 000111, \ldots$. Later, we will even *prove* that there cannot be a DFA for this language.

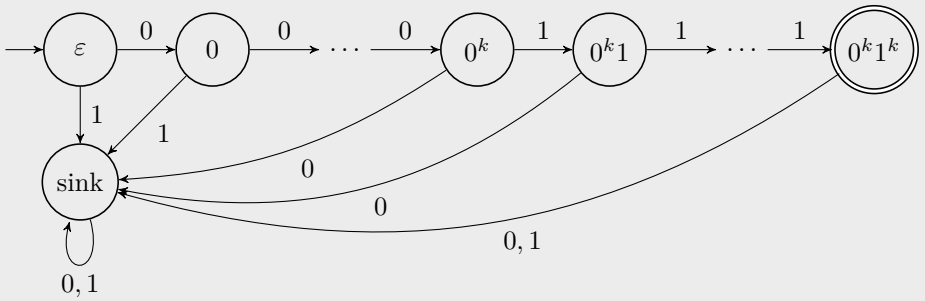
(i) DFA for L_1:



(ii) DFA for L_2:



(iii) Sketch of a construction of a DFA for L_k for a given k:



(iv) The problem is that a DFA for L would have to work for *every* k, and there are infinitely many. Using the same construction as above, this DFA needs to "count" the number of 0s in the prefix of the word it reads, and than needs to "check" whether the same number of 1s is found in the suffix of the word. If we follow this idea, however, we would construct an automaton of infinite size, and this contradicts the definition of a DFA.

1.3 Cycles in Finite Automata

Prove *by contradiction* that every DFA contains a cycle.

For a contradiction, let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA that does not contain any cycle. Let m denote the number of states of A, that is, $|Q| = m$. Consider one fixed letter $a \in \Sigma$. Since every DFA is complete by definition, there has to be a path P of length $m - 1$ from q_0 through A only following edges that are labeled with "a." Since A does not contain a cycle, P visits all states of A; it starts at q_0 and ends in some state q.

Again, since A is complete, there has to be an edge labeled "a" from q to some state q'. However, this state must have already been visited before on P, which is a direct contradiction to the assumption that A does not contain a cycle. Therefore, A contains a cycle.