Technische Universität München **Winter term 21/22**
I7
Prof. J. Esparza / A. R. Balasubramanian / C. Weil-Kennedy

Automata and Formal Languages — Exercise Sheet 1

You can find the Automata Tutor course name and password on the Moodle website for "Lecture and Exercises for Automata and Formal Languages (IN2041)". If you are enrolled for the course "Exercise - Automata and Formal Languages (IN2041)" in TUM online, you automatically have access to the Moodle website.

Exercise 1.1
Give a regular expression and a NFA for the language of all words over $\Sigma = \{a, b\}$ ...

1. ... with the second letter from the right being an a.
2. ... beginning with ab or ending with ba.
3. ... with at least one occurrence of a and at least one occurrence of b.
4. ... with no occurrence of the subword abb.

You can do this exercise in Automata Tutor to get feedback on your answer. The NFA constructions are under "NFA Construction" and called AFL 1.1 (i)*, while the regular expression are under "RE Construction" and called* AFL 1.1 (i) RE*.*

Exercise 1.2
Consider the language $L \subseteq \{a, b\}^*$ given by the regular expression b^*a^*ba.

1. Give an NFA that accepts L.
2. Give a DFA that accepts L.

You can do this exercise in Automata Tutor to get feedback on your answer. The first part is under "NFA Construction", the second is under "DFA Construction".

Exercise 1.3
Given $n \in \mathbb{N}_0$, let MSBF(n) be the set of *most-significant-bit-first* encodings of n, i.e., the words that start with an arbitrary number of leading zeros, followed by n written in binary. For example:

$$\text{MSBF}(3) = 0^*11 \quad \text{and} \quad \text{MSBF}(9) = 0^*1001 \quad \text{MSBF}(0) = 0^*.$$

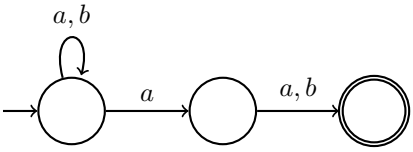
Similarly, let LSBF(n) denote the set of *least-significant-bit-first* encodings of n, i.e., the set containing for each word $w \in \text{MSBF}(n)$ its reverse. For example:

$$\text{LSBF}(6) = L(0110^*) \quad \text{and} \quad \text{LSBF}(0) = L(0^*).$$

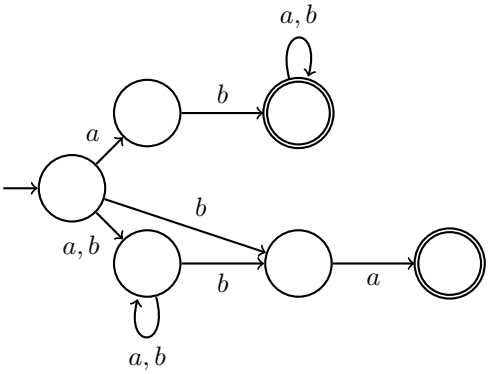
1. Construct and compare DFAs recognizing the encodings of the even numbers $n \in \mathbb{N}_0$ w.r.t. the unary encoding, where n is encoded by the word 1^n, and the MSBF-encoding, and the LSBF-encoding.
2. Same for the set of numbers divisible by 3, and for the set of numbers divisible by 4.
3. Give regular expressions corresponding to the languages in 2.

Solution 1.1
We write Σ for $(a+b)$ and Σ^* for $(a+b)^*$.

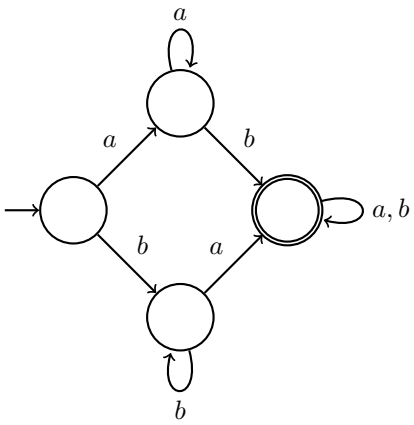
1. $\Sigma^* a \Sigma$



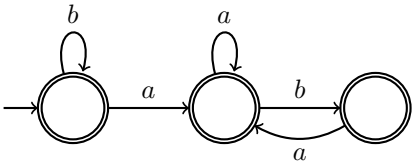
2. $ab\Sigma^* + \Sigma^* ba$



3. $aa^*b\Sigma^* + bb^*a\Sigma^*$

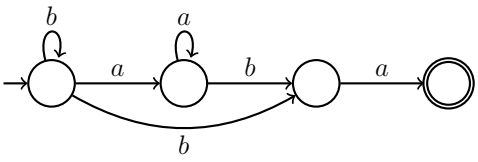


4. $b^*(ab+a)^*$

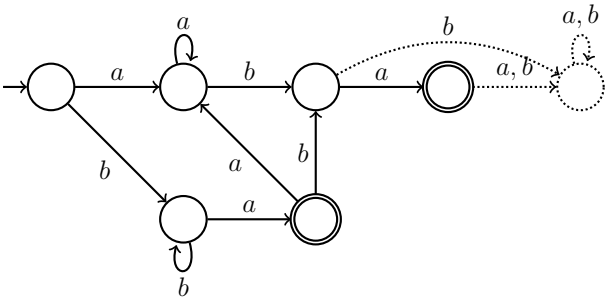


Solution 1.2

1. NFA accepting L:



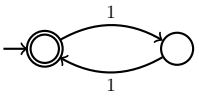
2. DFA accepting L:



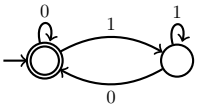
Solution 1.3

1. Here are the DFAs:

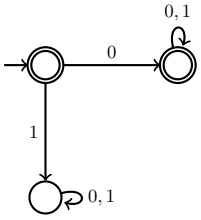
   - Unary encoding:



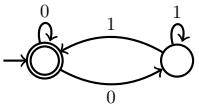
   - MSBF encoding. The DFA recognizes all strings ending with 0:



   - LSBF encoding. The DFA recognizes all strings starting with 0:



   Another way to obtain the same result is to observe that an automaton for the LSBF encoding must accept a word w iff the reverse of w is accepted by the DFA for the MSBF encoding. Using this observation we can easily construct an automaton for the LSBF encoding from the one for the MSBF encoding by just reversing the arcs (more generally, one also has to swap the initial and final states, but in this case this is not necessary, because the initial state is at the same time the only final state). This yields the automaton



   However, this is no longer a DFA, but an NFA. The DFA above is obtained by applying the *NFAtoDFA* algorithm.

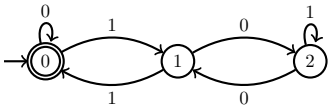
2.

   - *Numbers divisible by* 3*:*
     The DFA for the unary encoding is, loosely speaking, a circuit of length three. We now give a DFA for the MSBF encoding. The idea is that the state reached after reading the word u corresponds to the remainder of the number represented by u when dividing by 3. We therefore take as states $\{0, 1, 2\}$ with 0 as both

initial and final state. If a word w encodes a number k, then wa encodes the number $2k + a$. So for every state $q \in \{0, 1, 2\}$ we define

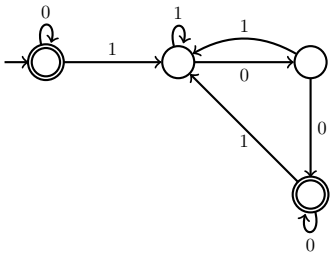
$$\delta(q, a) = 2q + a \pmod 3.$$

This yields the automaton:

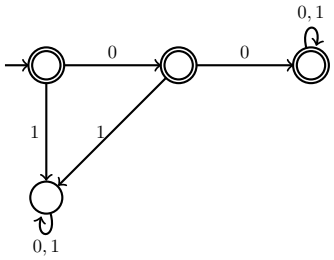


In order to obtain a DFA for the LSBF encoding we apply the same idea as above: exchange initial and final states, and reverse the arcs. In general, this yields an NFA, but in this case the result of this operation is the same automaton! So we have proved that a binary number $b_1 b_2 \dots b_n$ is divisible by 3 iff the number $b_n b_{n-1} \dots b_1$ is also divisible by 3.

- *Numbers divisible by* 4*:*
  The DFA for the unary encoding is, loosely speaking, a circuit of length four. We now give a DFA for the MSBF encoding. We can do the same as for numbers divisible by 3, with states for the remainders $\{0, 1, 2, 3\}$. Or we can simply rely on the fact that the numbers divisible by 4 are 0 or end with 00 in the MSBF encoding.



A DFA for the LSBF encoding is any DFA which accepts the empty word, the word 0 and any other word beginning with 00.



3. Regular expressions for

- The unary encodings are $(1^3)^*$ and $(1^4)^*$ respectively
- The MSBF encodings are $\big(0 + 1(01^*0)^*1\big)^*$ and $0^* + (0 + 1)^*00$ respectively.
- The LSBF encodings are $\big(0 + 1(01^*0)^*1\big)^*$ and $0^* + 00(0 + 1)^*$ respectively.