Problem Set for Tutorial 3 — TDDD14/TDDD85

1 DFA Minimization

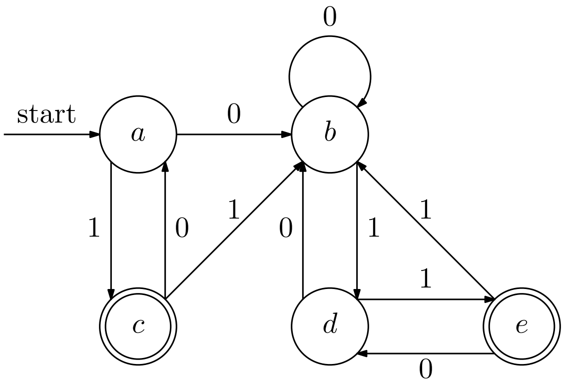


Figure 1: M_1

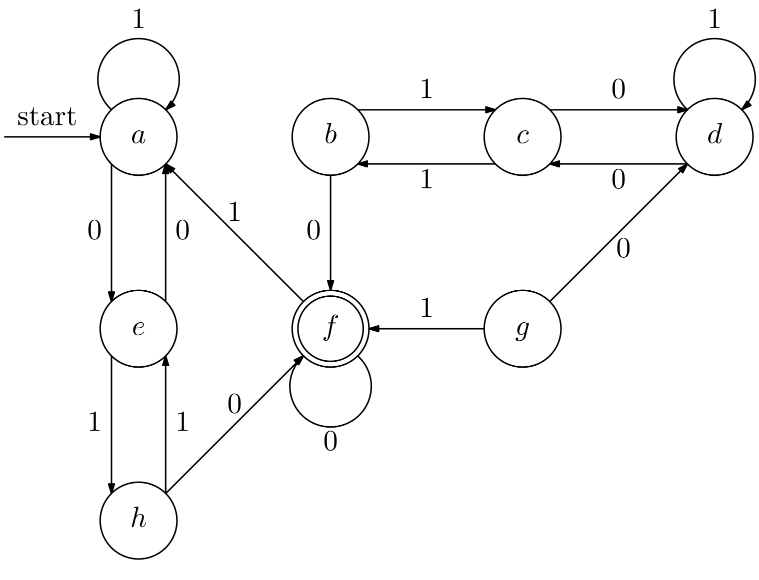


Figure 2: M_2.

Exercise 1.

1. Minimize the DFA in Figure 1.
2. Minimize the DFA in Figure 2.

Solutions

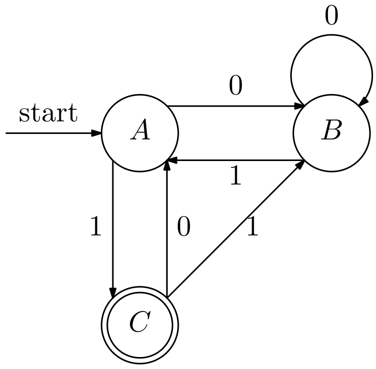


Figure 3: The DFA from Figure 1 minimized.

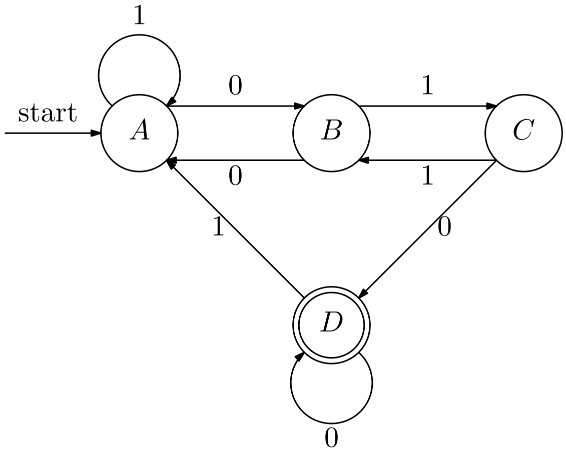


Figure 4: A DFA from Figure 2 minimized.

Solution to Exercise 1.

1. A minimal DFA is given in Figure 3.
2. A minimal DFA is given in Figure 4.

2 Regular Expressions

Exercise 2. Let $r = (1 + 00^*11)(0 + 1(0 + 10)^*11)^*$. Which of the following strings belong to $L(r)$?

1. 010001,
2. 00111011,
3. 1100110,
4. 101100,
5. 10011001.

Exercise 3. Give regular expressions for the following languages over the alphabet $\{0, 1\}$.

1. The set of all strings ending in 00.
2. The set of all strings in which the substring 00 occurs at most once.

Exercise 4. Construct an NFA_ε which accepts the language defined by the regular expression $10 + (0 + 11)0^*1$.

Exercise 5. Show that the equalities below hold for regular expressions. (r, s and t denote arbitrary regular expressions over some alphabet.)

1. $r + t = t + r$,
2. $r(s + t) = rs + rt$,
3. $(r + \epsilon)^* = r^*$,
4. $r\emptyset = \emptyset r = \emptyset$,
5. $\emptyset^* = \epsilon$.

Exercise 6. Give regular expressions that define

1. the language accepted by the DFA in Figure 5.
2. the language accepted by the DFA in Figure 6.

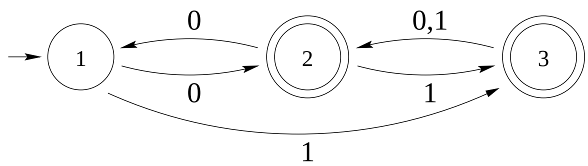


Figure 5: M_3

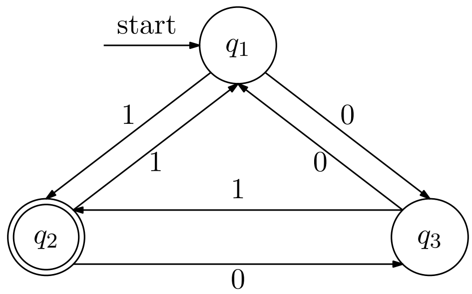


Figure 6: M_4

Solutions

Solution to Exercise 2.

1. 010001: no.
2. 00111011: yes.
3. 1100110: yes.
4. 101100: no.
5. 10011001: no.

Hint: no particular method is required here. Work systematically and use the fact that any string matched by the regular expression needs to consist of two substrings, one matching $(1 + 00^*11)$, and one matching $(0 + 1(0 + 10)^*11)$ zero or more times.

Solution to Exercise 3.

1. $(0 + 1)^*00$,
2. $(1 + 01)^*(\epsilon + 0 + 00)(1 + 10)^*$.

Solution to Exercise 4. By decomposing the regular expression syntactically according to the recursive definition of regular expressions, an NFA$_\varepsilon$ can be constructed systematically in a bottom-up fashion by successively joining NFA$_\varepsilon$s corresponding to subexpressions according to the regular operator (*, concatenation, +) in question. For details, consult the corresponding lecture manuscript. The resulting NFA$_\varepsilon$ is shown in Figure 7.

Solution to Exercise 5. For each pair of regular expressions x and y we need to verify that $L(x) = L(y)$. We provide detailed solutions to the first two exercises since the last three uses the same method.

1. $r + t = t + r$: $L(r + t) = L(r) \cup L(t) = L(t) \cup L(s)$ (the last equality follows from the fact that union is commutative, i.e., the order of the sets do not matter).
2. $r(s+t) = rs+rt$: $L(r(s+t)) = L(r)L(s+t) = L(r)(L(s) \cup L(t)) = L(r)L(s) \cup L(r) =$

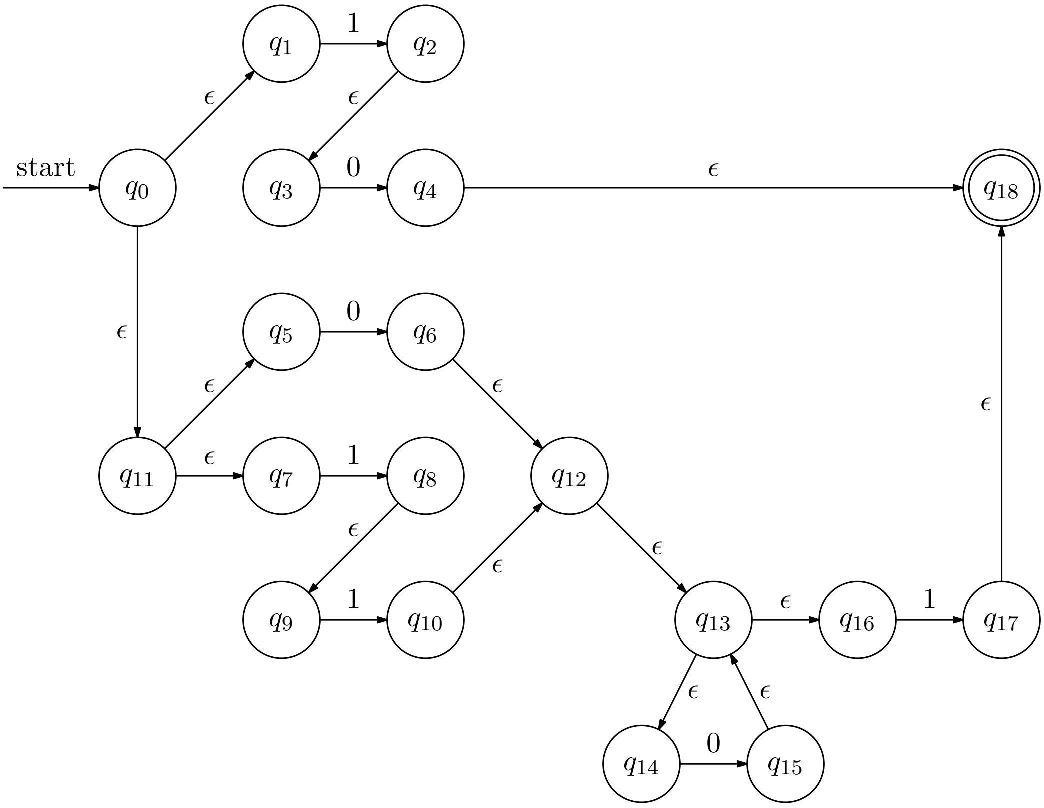
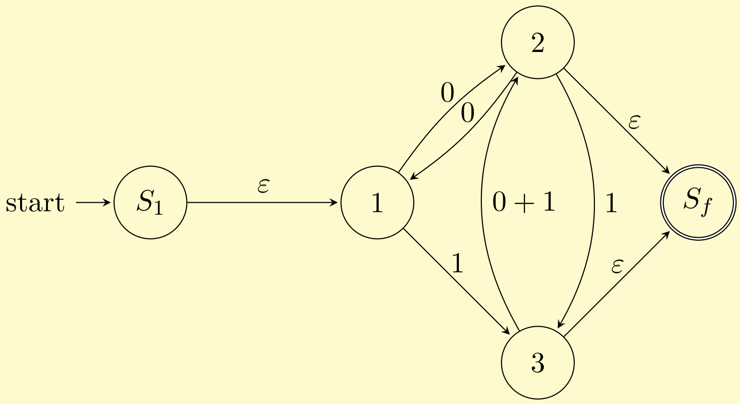


Figure 7: M_5

$L(r)L(t) = L(rs)L(rt)$ (here we use the fact that set concatenation distributes over union, i.e., for any sets A, B, C we have that $A(B \cup C) = AB \cup AC$).

Solution to Exercise 6. When presenting these solutions and simplifying a regular expression s to an equivalent regular expression t (i.e., $L(s) = L(t)$) we for simplicity write $s = t$ rather than $L(s) = L(t)$.

1. (a) Redraw and add a new start and final state to the GNFA.

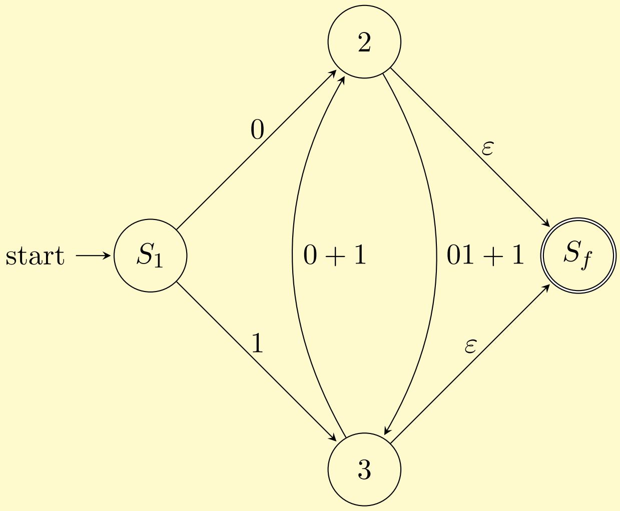


The state elimination steps are performed as follows.

(b) Eliminate state 1. There are four paths through state 1:

- $S_1 \to 1 \to 2 : \varepsilon\emptyset^*0 + \emptyset = 0$
- $S_1 \to 1 \to 3 : \varepsilon\emptyset^*1 + \emptyset = 1$
- $2 \to 1 \to 2 : 0\emptyset^*0 + \emptyset = 00$
- $2 \to 1 \to 3 : 0\emptyset^*1 + 1 = 01$

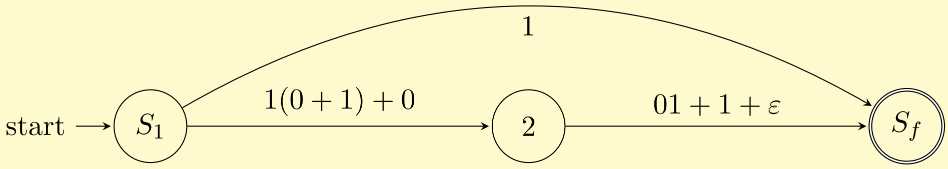
The GNFA now looks as follows.



(c) Eliminate state 3. There are four paths through state 3:

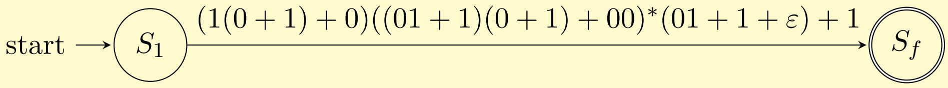
- $S_1 \to 3 \to 2 : 1\emptyset^*(0+1) + 0 = 1(0+1) + 0$
- $S_1 \to 3 \to S_f : 1\emptyset^*\varepsilon + \emptyset = 1$
- $2 \to 3 \to 2 : (01+1)\emptyset^*(0+1) + 00 = (01+1)(0+1) + 00$
- $2 \to 3 \to S_f : (01+1)\emptyset^*\varepsilon + \varepsilon = 01 + 1 + \varepsilon$[a].

The GNFA now looks as follows.



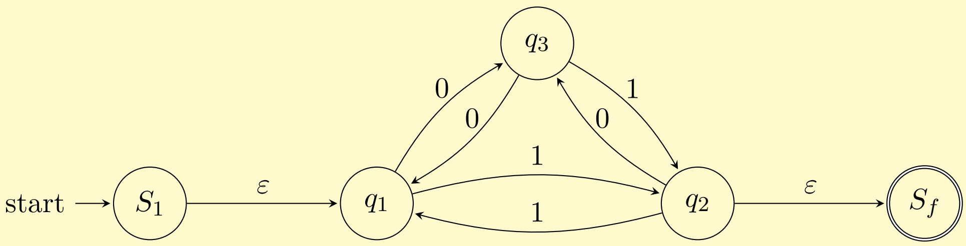
(d) Eliminate state 2. There is one path from state S to state Sf:

- $S_1 \to 2 \to Sf : (1(0+1)+0)((01+1)(0+1)+00)^*(01+1+\varepsilon)+1$.



This regular expression describes the language of the original DFA.

2. Redraw and add a new start and final state to the GNFA.

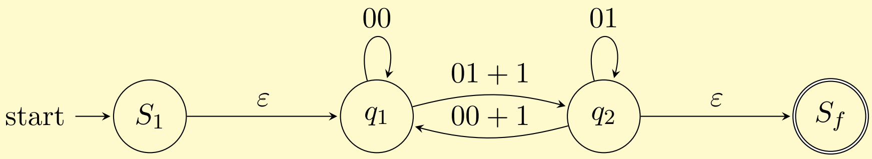


The state elimination steps are performed as follows.

3. Eliminate state q_3. There are four paths through state q_3:

   - $q_1 \to q_3 \to q_1 : 0\emptyset^*0 + \emptyset = 00$
   - $q_1 \to q_3 \to q_2 : 0\emptyset^*1 + 1 = 01 + 1$
   - $q_2 \to q_3 \to q_1 : 0\emptyset^*0 + 0 + 1 = 00 + 1$
   - $q_2 \to q_3 \to q_2 : 0\emptyset^*1 + \emptyset = 01$

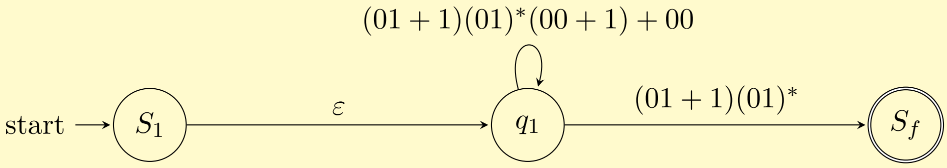
   The GNFA now looks as follows.



4. Eliminate state q_2. There are two paths through state q_2:

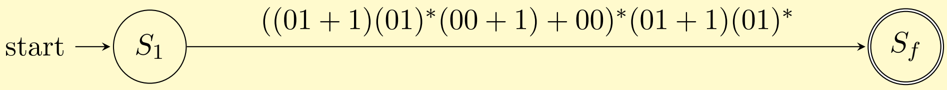
   - $q_1 \to q_2 \to q_1 : (01 + 1)(01)^*(00 + 1) + 00$
   - $q_1 \to q_2 \to S_f : (01 + 1)(01)^*\varepsilon + \emptyset = (01 + 1)(01)^*$

   The GNFA now looks as follows.



5. Eliminate state q_1. There is only one path.

   - $S_1 \to q_1 \to Sf : \varepsilon((01+1)(01)^*(00+1)+00)^*(01+1)(01)^*+\emptyset = ((01+1)(01)^*(00+1) + 00)^*(01 + 1)(01)^*$



   This regular expression describes the language of the original DFA.

[a]Note in particular that we in this case *cannot* simplify this to $01 + 1$.

3 Advanced and Exam Like Exercises

Exercise 7. Let the languages L_1 and L_2 be defined as follows:

- L_1 is defined by the regular expression $(a + b)^*bba(a + b)^*$.
- L_2 is the language of strings over $\{a, b\}^*$ containing the string ab.

Give a regular expression R such that $L(R) = L_1 - L_2$, *i.e*, $L(R) = \{w \mid w \in L_1 \land w \notin L_2\}$. Explain your reasoning and why your solution is correct.

Exercise 8. Using a standard method, construct a regular expression defining the same language as the DFA whose transition function δ is given by

	a	b
$\rightarrow A$	A	C
B F	A	B
C F	B	A

Exercise 9. For each pair of regular expressions R_1 and R_2 below, answer whether they generate the same language ($L(R_1) = L(R_2)$). If no, give a string which belongs to one of the languages and does not belong to the other. If yes, show that they are equivalent, e.g., by (1) computing $L(R_1)$ and $L(R_2)$ as far as you can and (2) verifying that the two resulting sets are equal. For the last step, an informal explanation is sufficient.

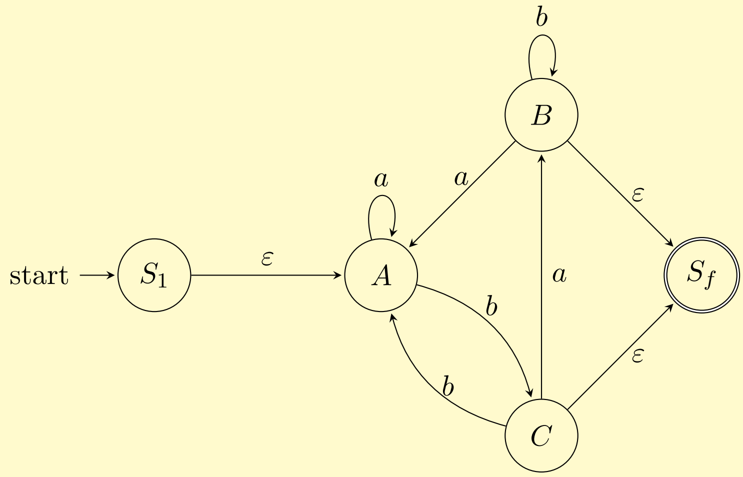
1. $\varepsilon + \varepsilon$ and ε.
2. $\emptyset + \emptyset$ and $\emptyset$.
3. $a(b + c + \varepsilon)$ and $ab + bc$.
4. $(ab + a)^*a$ and $a(ba + a)^*$.

Solutions

Solution to Exercise 7. Hint: while it is possible to solve the problem by a systematic approach by constructing a DFA for $L_1 \cap \bar{L_2}$ and converting this DFA to a regular expression, it is *much* easier to construct the regular expression directly. Hence, how can you adapt the regular expression $(a+b)^*bba(a+b)^*$ so that it does not match ab?

Solution to Exercise 8.

1. Redraw and add a new start and final state to the GNFA.

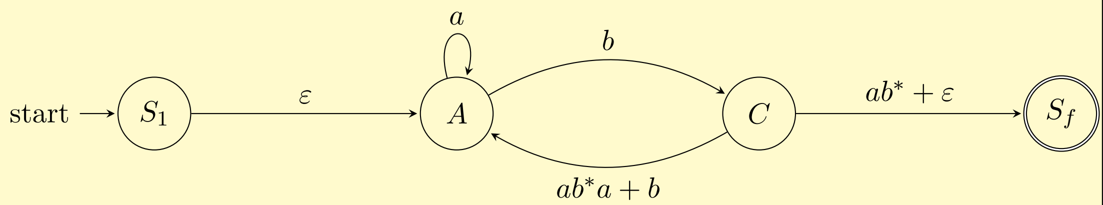


   The state elimination steps are performed as follows.

2. Eliminate state B. There are 2 paths through state B:

   - $C \to B \to A : ab^*a + b$
   - $C \to B \to S_f : ab^*\varepsilon + \varepsilon = ab^* + \varepsilon$

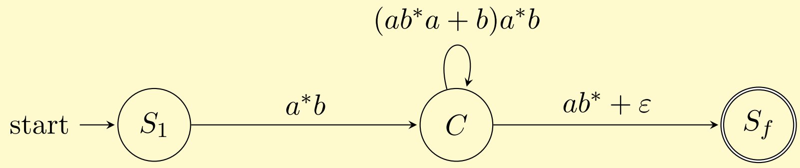
   The GNFA now looks as follows.



3. Eliminate state A. There are two paths through state A:

   - $S_1 \to A \to C : \varepsilon a^*b + \emptyset = a^*b$
   - $C \to A \to C : (ab^*a + b)a^*b + \emptyset = (ab^*a + b)a^*b$

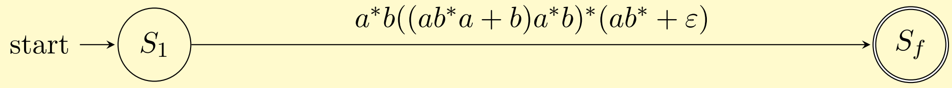
   The GNFA now looks as follows.



4. Eliminate state C.

   - $S_1 \to C \to Sf : a^*b((ab^*a+b)a^*b)^*(ab^*+\varepsilon)+\emptyset = a^*b((ab^*a+b)a^*b)^*(ab^*+\varepsilon)$.

   We obtain the following GNFA.



This regular expression describes the language of the original DFA.

Solution to Exercise 9.

1. $L(\varepsilon+\varepsilon) = L(\varepsilon) \cup L(\varepsilon) = \{\varepsilon\} \cup \{\varepsilon\} = \{\varepsilon\}$. Hence, they are the same.

2. $\emptyset + \emptyset$ and $\emptyset$: a similar argument to the above shows that they are the same.

3. $L(a(b+c+\varepsilon)) = L(a)L(b+c+\varepsilon) = \{a\}\{a,b,\varepsilon\} = \{aa,ab,a\}$ which is not the same as $L(ab+bc) = \{ab,bc\}$.

4. $L((ab+a)^*a) = \{ab,a\}^*\{a\}$, i.e., the set of strings starting with an arbitrary combination of ab and a and ending with a. For the other expression we see that $L(a(ba+a)^*) = \{a\}\{ba,a\}^*$, i.e., the set of strings starting with an a and ending with an arbitrary combination of ba or a. Since a concatenated with ba gives us the same as ab concatenated with a it is not hard to see that the two expressions describe the same language.