Montogue

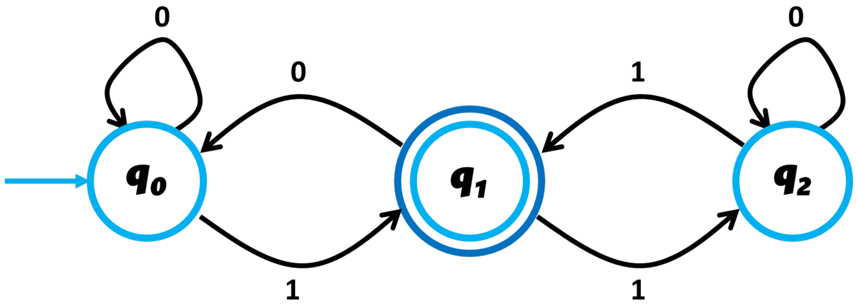
Quiz EL501
Finite Automata and Regular Expressions

Lucas Monteiro Nogueira

PROBLEMS

Problem 1

Which of the strings 01101 and 0000110011 are accepted by the deterministic finite automaton illustrated below?



Problem 2

For an alphabet $\Sigma = \{a,b\}$, construct a DFA:

Problem 2.1: that accepts all strings with exactly one a.
Problem 2.2: that accepts all strings with at least one a.
Problem 2.3: that accepts all strings with no more than three a's.
Problem 2.4: that accepts all strings with at least one a and exactly two b's.
Problem 2.5: that accepts all strings that start with an a and have at most one b.
Problem 2.6: that accept all strings that have an odd number of a's and end with a b.

Problem 3

Give state diagrams of DFA's recognizing the following languages. The alphabet is {0,1}.

Problem 3.1: $\{w \mid w$ begins with a 1 and ends with a 0$\}$.
Problem 3.2: $\{w \mid w$ has length at least 3 and its third symbol is a 0$\}$.
Problem 3.3: $\{w \mid w$ does not contain the substring 110$\}$.
Problem 3.4: $\{w \mid w$ is any string except 11 and 111$\}$

Problem 4

Draw up transition tables for the following for DFA's accepting the following languages over the alphabet {0,1}:

Problem 4.1: The set of all strings ending in 00.
Problem 4.2: **(Difficult)** The set of all strings beginning with a 1 that, when interpreted as a binary integer, is a multiple of 5. For example, strings 101, 1010, and 1111 are in the language; 0, 100, and 111 are not.

Problem 5

Consider the following ε-nondeterministic finite automaton. (As usual, an arrow indicates the initial state, and an asterisk denotes an acceptable final state.)

	ε	a	b	c
$\rightarrow p$	$\emptyset$	$\{p\}$	$\{q\}$	$\{r\}$
q	$\{p\}$	$\{q\}$	$\{r\}$	$\emptyset$
$*r$	$\{q\}$	$\{r\}$	$\emptyset$	$\{p\}$

Problem 5.1: Compute the ε-closure of each state.
Problem 5.2: Give all the strings of length three or less accepted by the automaton.

Problem 6

Problem 6.1: The formal description of a DFA $\mathcal{M}_1$ is $(\{q_0, q_1, q_2, q_3\}, \{0,1\}, \delta, q_0, \{q_1,q_2\})$, where δ is given by the following table. Give the state diagram of this machine. Does this machine accept the string 000? What about the string 010?

	0	1
q_0	q_1	q_3
q_1	q_2	q_3
q_2	q_2	q_2
q_3	q_3	q_3

Problem 6.2: The formal description of a DFA $\mathcal{M}_2$ is $(\{q_1, q_2, q_3, q_4, q_5\}, \{u,d\}, \delta, q_3, \{q_3\})$, where δ is given by the following table. Give the state diagram of this machine.

	u	d
q_1	q_1	q_2
q_2	q_1	q_3
q_3	q_2	q_4
q_4	q_3	q_5
q_5	q_4	q_5

Problem 7

Give regular expressions for the following languages.

Problem 7.1: $L_1 = \{a^n b^m : n \geq 4, m \leq 3\}$

Problem 7.2: $L_2 = \{a^n b^m : n \geq 3, m \text{ is even}\}$

Problem 7.3: $L_3 = \{a^n b^m : (n+m) \text{ is even}\}$

Problem 7.4: Give a simple verbal description of language L_4, namely

$$L_4 = ((aa)^* b(aa)^* + a(aa)^* ba(aa)^*)$$

Problem 8

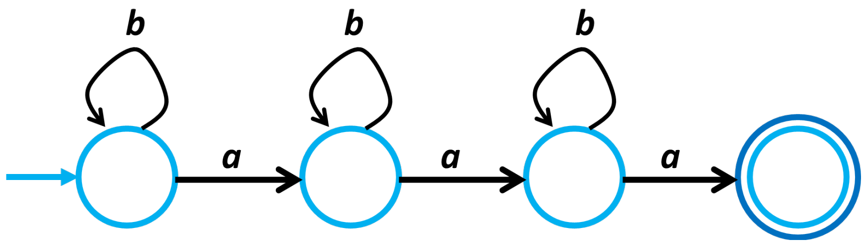
Give regular expressions for the following languages on $\Sigma = \{a, b, c\}$.

Problem 8.1: All strings containing exactly one a.
Problem 8.2: All strings containing no more than three a's.
Problem 8.3: All strings that contain at least one occurrence of each symbol in Σ.
Problem 8.4: All strings that contain no run of a's of length greater than two.
Problem 8.5: Find a regular expression for the language accepted by automaton $\mathcal{M}_1$ of Problem 6.1.

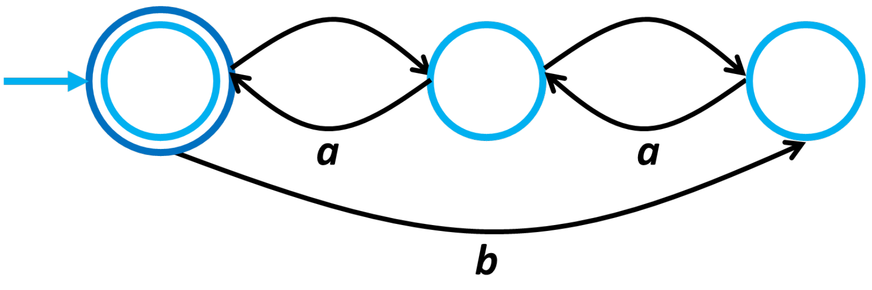
Problem 9

Find a regular expression for the languages accepted by the following automata.

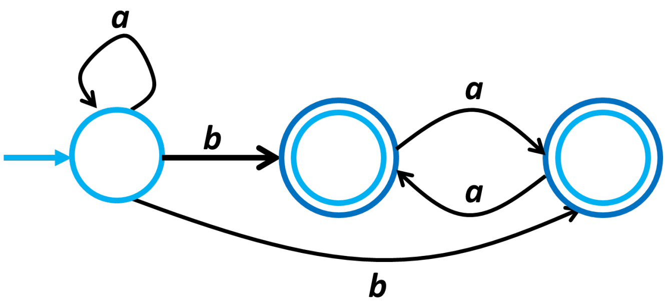
Problem 9.1:



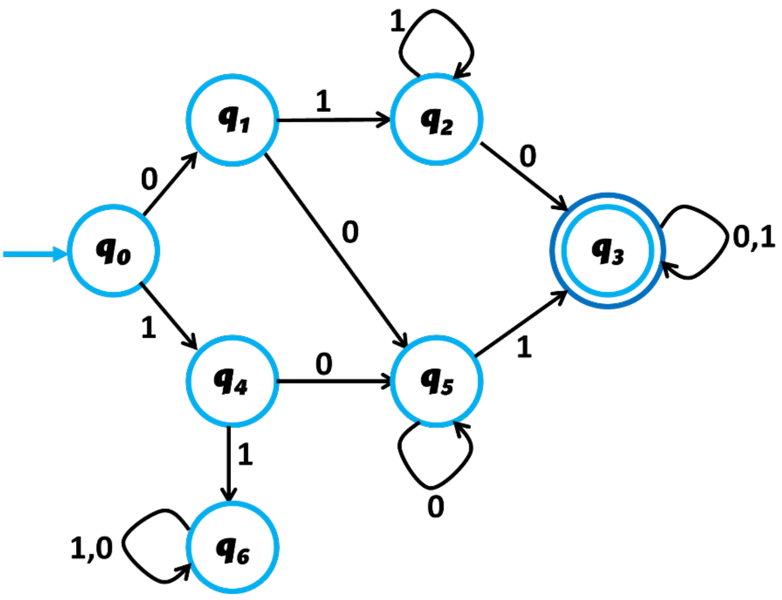
Problem 9.2:



Problem 9.3:

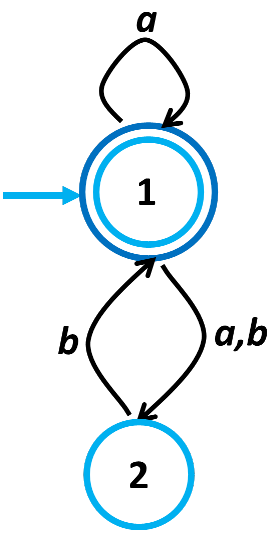


Problem 9.4:



Problem 10

Problem 10.1: Convert the following nondeterministic finite automaton to an equivalent deterministic finite automaton.



Problem 10.2: Convert to a DFA the NFA described by the following transition table. (Recall that an arrow denotes a start state, and an asterisk denotes an accept state.)

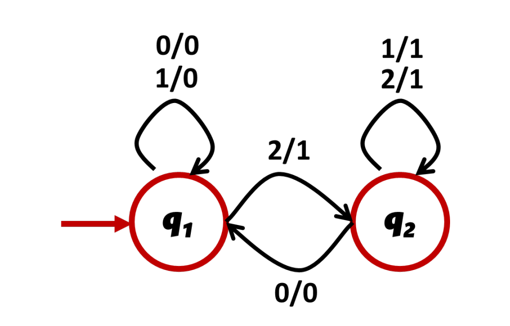
	0	1
$\to p$	$\{p,q\}$	$\{p\}$
q	$\{r\}$	$\{r\}$
r	$\{s\}$	$\emptyset$
$*s$	$\{s\}$	$\{s\}$

Problem 11 (Sipser, 2013)

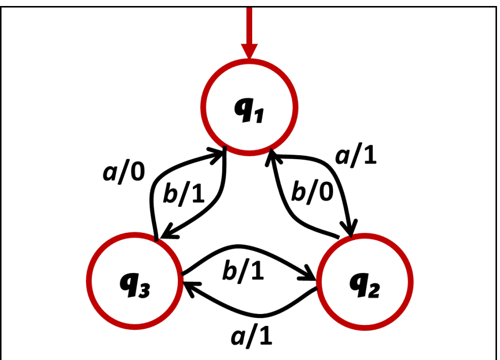
A finite state transducer (FST) is a type of deterministic finite automaton whose output is a string and not just *accept* or *reject*. The following are state diagrams of finite state transducers T_1 and T_2.

Each transition of an FST is labeled with two symbols, one designating the input symbol for that transition and the other designating the output symbol. The two symbols are written with a slash, /, separating them. In T_1, the transition from q_1 to q_2 has input symbol 2 and output symbol 1. Some transitions may have multiple input-output pairs, such as the transition in T_1 from q_1 to itself. When an FST computes on an input string w, it takes the input symbols $w_1 \dots w_n$ one by one and, starting at the start state, follows the transitions by matching the input labels with the sequence of symbols $w_1 \dots w_n = w$. Every time it goes along a transition, it outputs the corresponding

output symbol. For example, on input 2212011, machine T_1 enters the sequence of states $q_1, q_2, q_2, q_2, q_2, q_1, q_1, q_1$ and produces output 1111000. On input *aabb*, T_2 outputs 1011. Give the sequence of states entered and the output produced in each of the following parts.
Problem 11.1: T_1 on input 2100
Problem 11.2: T_2 on input *abbba*



Finite state transducer T_1



Finite state transducer T_2

SOLUTIONS

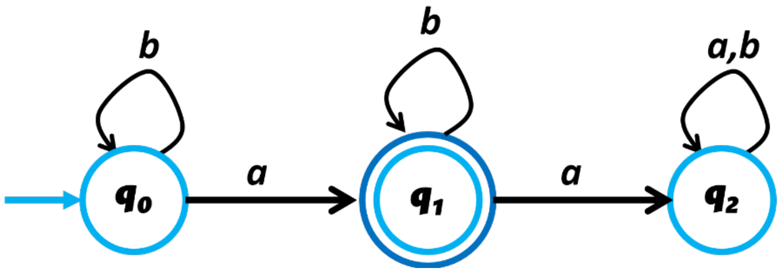
P.1 ➡ *Solution*

Consider first string 01101. Starting at q_0, we see that an input of 0 causes the accepter to remain at q_0. Then, an input of 1 causes the accepter to change from state q_0 to q_1. A second input of 1 switches the accepter from vertex q_1 to q_2. Then, by taking an input of 0 the system remains at q_2. Lastly, the accepter takes an input of 1 to transition from q_2 to q_1. Since the DFA ends at q_1, the string is indeed accepted by the DFA at hand.

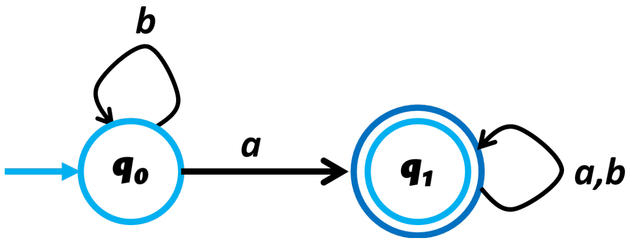
Proceeding similarly with the string 0000110011, it can be shown that when processing such a string the DFA would end up at state q_2, which is not the final state. Thus, the string in question is *not* accepted by the system.

P.2 ➡ *Solution*

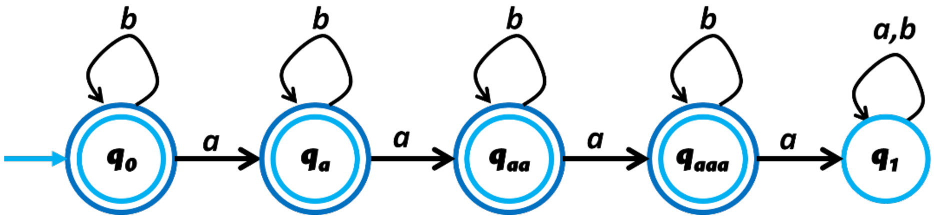
Problem 2.1: The DFA below describes an accepter that takes an input of exactly one a. Notice that, starting at q_0, an input a causes the system to switch to state q_1, which is the final state. Similarly, an input, say, abb, which also contains one a, would ultimately leave the DFA at final state q_1. The accepter will remain in the final state so long as the input contains exactly one a, as desired.



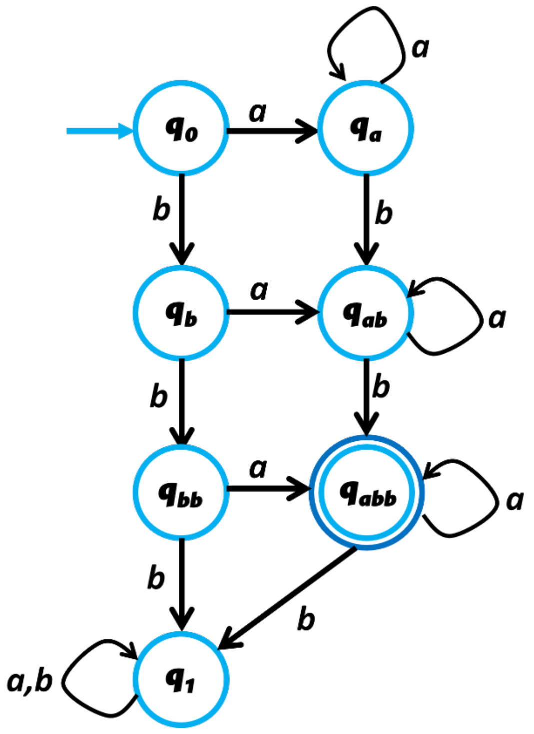
Problem 2.2: The DFA below describes an accepter that takes inputs containing at least one a. Starting at q_0, an input a will cause the accepter to migrate from q_0 to q_1, which is the final state. Once at q_1, any input, be it a or b, will not transition back to q_0; that is, q_1 functions as a trap state.



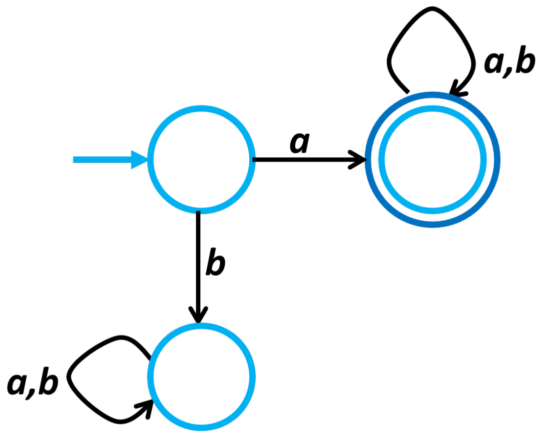
Problem 2.3: The simplest way to design a DFA that accepts no more than three a's is to resort to a system with multiple final states, as shown.



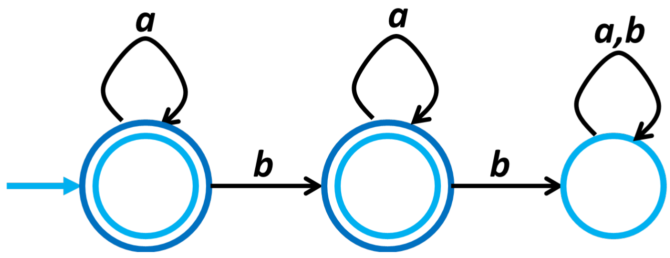
Problem 2.4: The DFA in question is illustrated below.



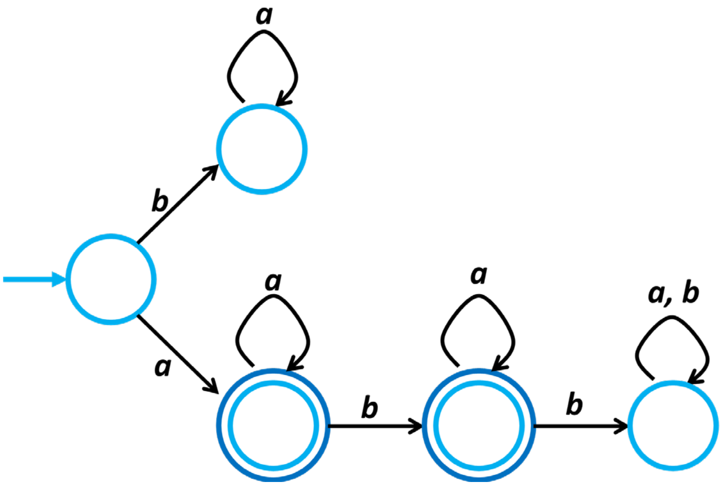
Problem 2.5: Consider the language L = {w | w starts with an a and has at most one b}. The language L is the intersection of two simpler languages L_1 and L_2 such that L_1 = {w | w starts with a} and L_2 = {w | w has at most one b}. The state diagram of the dfa that accepts L_1 is sketched below.



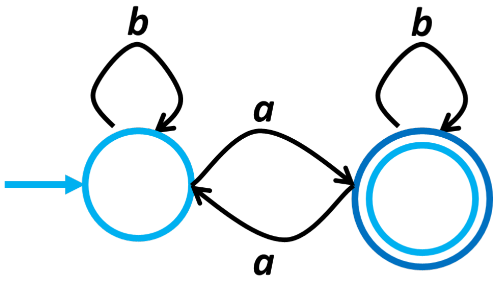
Next, we sketch the state diagram of the DFA that accepts L_2.



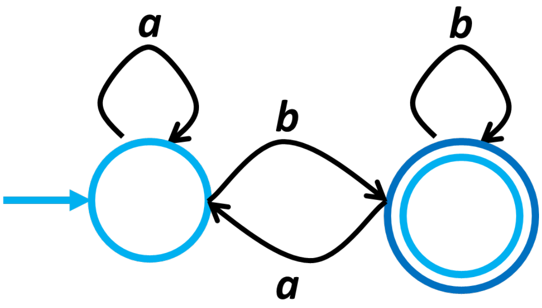
The state diagram of the machine that accepts the intersection of L_1 and L_2 is drawn next.



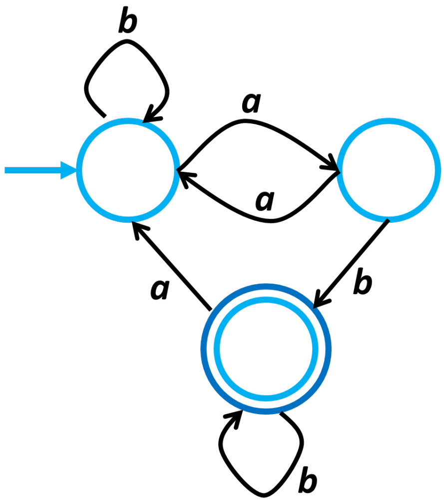
Problem 2.6: Consider the language L = {w | w has an odd number of a's and ends with a b}. The language L is the intersection of two simpler languages L_1 and L_2 such that L_1 = {w | w has an odd number of a's} and L_2 = {w | w ends with a b}. The state diagram of the DFA that accepts L_1 is sketched below.



We proceed to draw the state diagram for the DFA that accepts L_2.

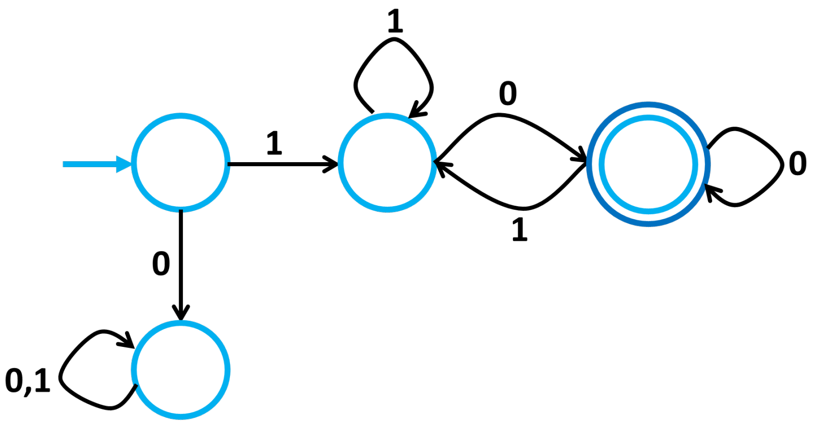


It remains to sketch the state diagram of the machine that accepts the intersection of L_1 and L_2, as shown.

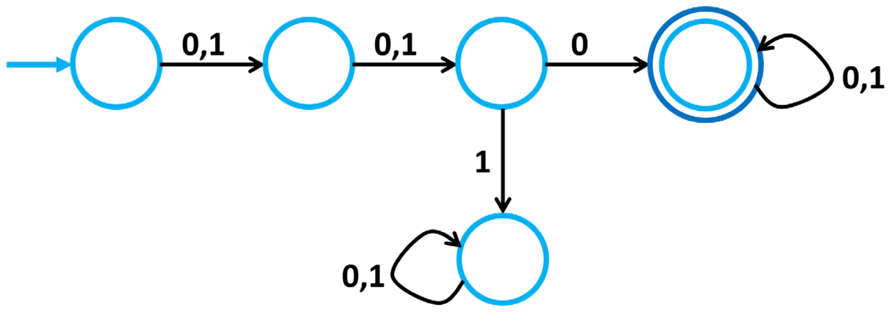


P.3 ➡ *Solution*

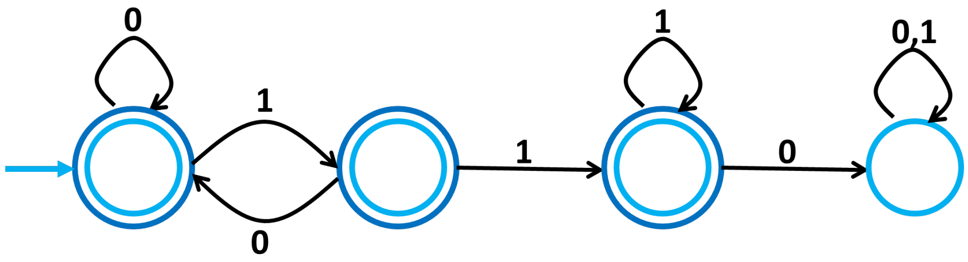
Problem 3.1: The state diagram is sketched below. Note that, starting at the initial state, an input 0 causes the automaton to switch to the state represented by the lowermost node, which is a trap state. This is appropriate, since the system is supposed to accept only inputs that begin with a 1. If, however, the first input is a 1, the automaton switches to a vertex to the right of the initial state. Once there, the accepter will loop in the same state whenever the input is a 1, or it can be displaced to the rightmost state if the input is a zero. The rightmost state is an acceptable final state, because the machine is intended to accept inputs that end in zero.



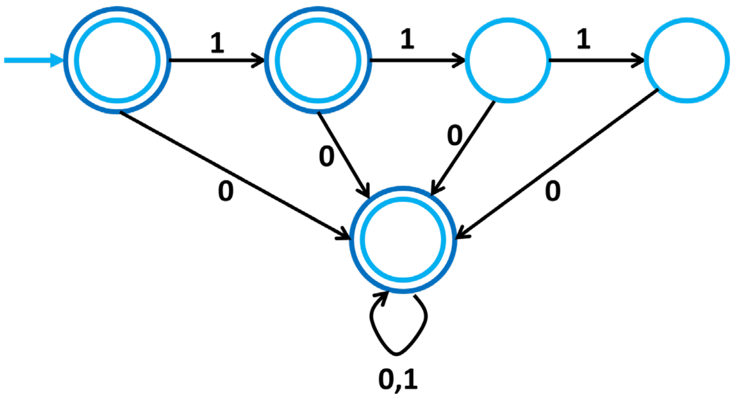
Problem 3.2: The state diagram in question is shown on the next page. Beginning at the initial state, the automaton transitions along three vertices, all of which are not acceptable final states because the length of *w* must be at least 3. At the third such node, taking a 1 will displace the system to a trap state, because the third symbol of *w* should be a 0; if the system takes a zero at the third node, an acceptable final state will be reached.



Problem 3.3: The state diagram in question is shown below. This one is relatively easy because the automaton can take in most states, except those that contain the substring 110.



Problem 3.4: The state diagram in question is shown below.



P.4 ➡ *Solution*

Problem 4.1: In the following table, states A, B, and C mean that the string seen so far ends in 0, 1, or 2 zeros. The arrow indicates the start state, and the star indicates an accepting state.

	0	1
$\rightarrow A$	B	A
B	C	A
$*C$	C	A

Problem 4.2: The trick is to realize that reading another bit either multiplies the number seen so far by 2 (if it is a 0), or multiplies by 2 and then adds a 1 (if it is a 1). We don't need to remember the entire number seen, just its remainder when divided by 5. That is, if we have any number of the form $5a+b$, where b is the remainder, between 0 and 4, then $2(5a+b) = 10a + 2b$. Since $10a$ is surely divisible by 5, the remainder of $10a + 2b$ is the same as the remainder of $2b$ when divided by 5. Since b is 0, 1, 2, 3, or 4, we can easily tabulate the answers. The same idea holds if we want to consider what happens when $5a+b$ if we multiply by 2 and add 1. The following table describes this automaton. State q_n means that the input seen so far has remainder n when divided by 5. As before, an arrow indicates a start state, and a star indicates an accepting state.

,

	0	1
$\rightarrow * q_0$	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_0
q_3	q_1	q_2
q_4	q_3	q_4

There is a small matter, however, that this automaton accepts strings with leading 0's. Since the problem asks for accepting only those strings that begin with a 1, we need an additional state ς, the start state, and an additional "dead state" d. If, in state ς, we see a 1 first, we act like state q_0, i.e., we go to state q_1. However, if the first input is 0, we should never accept, so we go to state d, which we never leave. The complete automaton is described below.

	0	1
$\rightarrow \varsigma$	d	q_1
$*\ q_0$	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_0
q_3	q_1	q_2
q_4	q_3	q_4
d	d	d

P.5 ➡ *Solution*

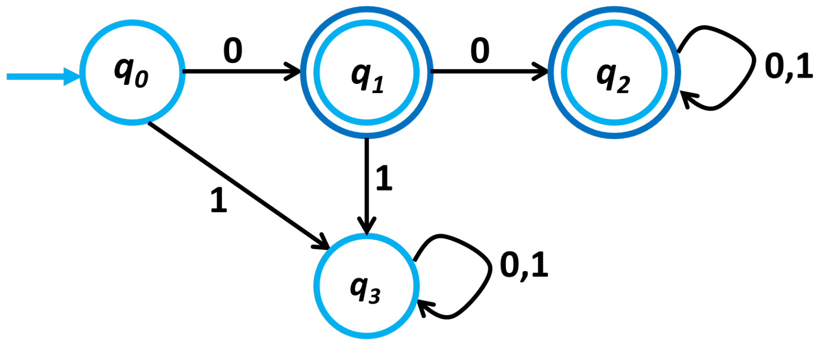
Problem 5.1: The closure of p is just $\{p\}$; for q it is $\{p,q\}$; and for r it is $\{p,q,r\}$.

Problem 5.2: Begin by noticing that a always leaves the state unchanged. Thus, we can think of the effect of strings of b's and c's only. To begin, notice that the only ways to get from p to r for the first time using only b, c and ε-transitions are bb, bc, and c. After getting to r, we can return to r by reading either b or c. Thus, every string of length 3 or less, consisting of b's and c's only, is accepted, with the exception of the string b. However, we have to allow a's as well. When we try to insert a's in these strings, yet keeping the length to 3 or less, we find that every string of a's b's and c's with at most one a is accepted. Also, the strings consisting of one c and up to two a's are accepted; other strings are rejected. There are three DFA states accessible from the initial state, which is the closure of p, or $\{p\}$. Let $A = \{p\}$, $B = \{p,q\}$, and $C = \{p,q,r\}$. Then the transition table is (an arrow denotes a starting state; an asterisk denotes an acceptable final state):

	a	b	c
$\rightarrow A$	A	B	C
B	B	C	C
$*C$	C	C	C

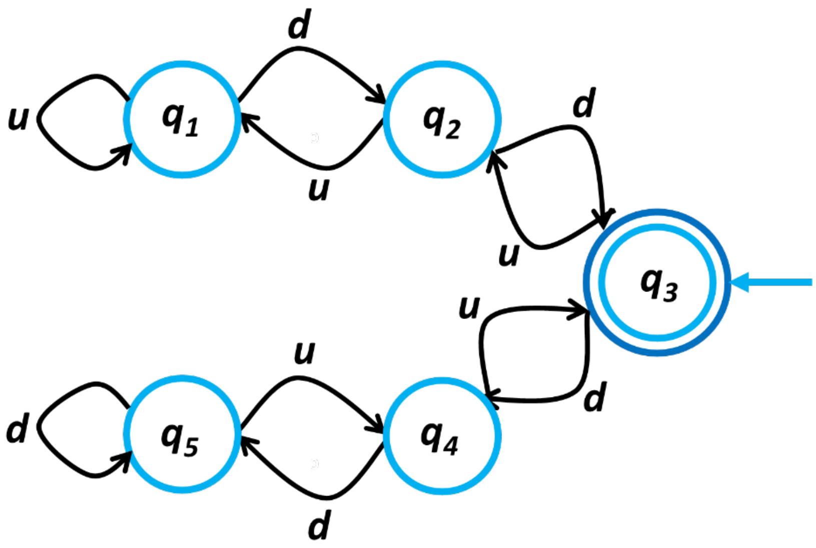
P.6 ➡ *Solution*

Problem 6.1: The transition diagram of $\mathcal{M}_1$ is shown below.



By taking the string 000, the automaton will respond with the state transition sequence $q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_2$; since q_2 is a viable final state, we conclude that the machine can accept the string 000. Upon taking the string 010, the automaton will exhibit the state transition sequence $q_0 \rightarrow q_1 \rightarrow q_3 \rightarrow q_3$; noting that q_3 is *not* a viable final state, we surmise that the machine will reject the string 010.

Problem 6.2: The state diagram is shown below.



P.7 ➡ *Solution*

Problem 7.1: Since we are to take a to $n \geq 4$, part of the regex we're looking for reads as

$$aaaaa^*$$

Further, b is taken to m less than or equal to 3, so we may write

$$(\varepsilon + b + bb + bbb)$$

This leads to the regex

$$\underline{r_1 = aaaa^*(\varepsilon + b + bb + bbb)}$$

Problem 7.2: For even m, b^m can be represented as

$$(bb)^*$$

For $n \geq 3$, a^n becomes

$$aaaa^*$$

The regex we're looking for is then

$$\underline{r_2 = aaaa^*(bb)^*}$$

Problem 7.3: For $n + m$ to be even, there are two possibilities: (I) both n and m are even; or (II) both n and m are odd. In case (I), noting that for even n and m, then $a^n = (aa)^*$ and $b = (bb)^*$, we may write

$$a^n b^m = (aa)^*(bb)^*$$

In case (II), noting that for odd n and m, then $a^n = (aa)^*a$ and $b^n = (bb)^*b$, giving

$$a^n b^m = (aa)^* a (bb)^* b$$

Finally, the regex we're looking for is

$$\underline{r_3 = (aa)^*(bb)^* + (aa)^* a (bb)^* b}$$

Problem 7.4: The language describes strings of the form $w_1 b w_2$, where w_1 and w_2 are composed of an even number of a's, or w_1 and w_2 consist of an odd number of a's.

P.8 ➡ *Solution*

Problem 8.1: The regex we aim for is

$$\underline{r = (b+c)^* a (b+c)^*}$$

Problem 8.2: The regex we aim for is

$$\underline{\begin{gathered} r = (b+c)^* + (b+c)^* a (b+c)^* \\ + (b+c)^* a (b+c)^* a (b+c)^* + (b+c)^* a (b+c)^* a (b+c)^* a (b+c)^* \end{gathered}}$$

Problem 8.3: The regex we aim for is

$$\underline{r = (a+b+c)^* a (a+b+c)^* b (a+b+c)^* c (a+b+c)^*}$$

Problem 8.4: The regex we aim for is

$$\underline{r = (\varepsilon + a + aa + b + c)^*}$$

Problem 8.5: We observe that once the DFA $\mathcal{M}_1$ enters state q_3, it will never leave this state and the input will be rejected regardless of the string entered; thus, q_3 is a trap state. On the other hand, once $\mathcal{M}_1$ enters state q_2, it will likewise remain in this state and the input will be accepted. From these observations, we conclude that $L(\mathcal{M}_1)$ consists of the string 0 (which ends at state q_1) and all strings starting with 00. In regular expression notation, we have

$$\underline{L(M_1) = 0 + 00(0+1)^*}$$

P.9 ➡ *Solution*

Problem 9.1: The language accepted by the automaton is $L = \{aaa$, $baaa$, $aaaa$, $aaba$, $baaaba$, ...$\}$. The corresponding regex is

$$\underline{L = b^* aa^* ab^* a}$$

Problem 9.2: The language accepted by this automaton is L = {ε, ab, bbb, $aabb$, $aababb$, $bbabb$, ...}. The corresponding regex is

$$\underline{L = \left(ab + \left(aa + b\right)\left(ba\right)^{*}bb\right)}$$

Problem 9.3: The language accepted by this automaton is L = {b, ab, ba, baa, aab, $baaaaa$, ...}. The corresponding regex is

$$\underline{L = a^{*}ba^{*}}$$

Problem 9.4: Clearly, q_6 is an unacceptable trap state, and q_3 is a success trap state. Thus, the problem boils down to finding how we can reach q_3 from the starting state q_0. It is easy to see that there are three kinds of paths linking q_0 to q_3:

$$(q_0, q_1, q_2, \ldots, q_2, q_3)$$
$$(q_0, q_1, q_5, \ldots, q_5, q_3)$$
$$(q_0, q_4, q_5, \ldots, q_5, q_3)$$

These paths correspond to strings beginning with 011*0, 000*1, and 100*1, respectively. Accordingly, the language accepted by the automaton is

$$\underline{L = \left(011^{*}0 + 000^{*}1 + 100^{*}1\right)\left(0 + 1\right)^{*}}$$

P.10 ➡ *Solution*

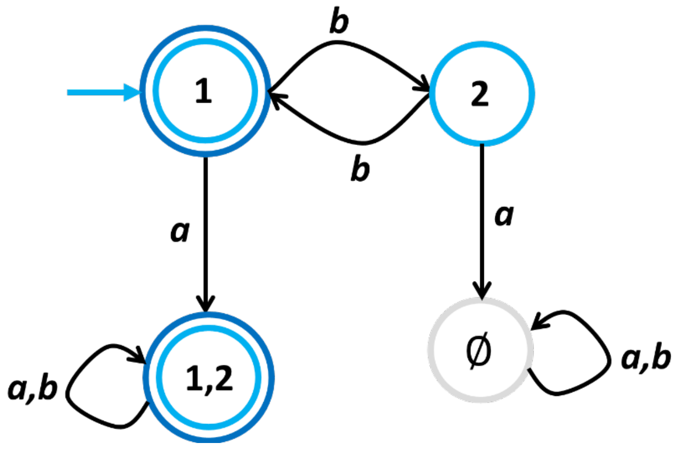
Problem 10.1: To construct an equivalent DFA from a given NFA, we begin by writing $Q' = P(Q)$, where $P(Q)$ is the set of subsets of state set Q. In the present case, Q' = {∅, {1}, {2}, {1,2}}. For an element R in Q' and a in set of alphabets Σ, we calculate

$$\delta'\left(R,a\right) = \{q \in Q \mid q \in \delta\left(r,a\right) \text{for some } r \in R\}$$

$\delta'(R,a)$ performs the transition on r for some value of a. Proceeding with the computation of transition function values, we have

$$\delta'\left(\varnothing,a\right) = \delta\left(\varnothing,a\right) = \varnothing$$
$$\delta'\left(\varnothing,b\right) = \delta\left(\varnothing,b\right) = \varnothing$$
$$\delta'\left(\{1\},a\right) = \delta\left(1,a\right) = \{1,2\}$$
$$\delta'\left(\{2\},a\right) = \delta\left(2,a\right) = \phi$$
$$\delta'\left(\{2\},b\right) = \delta\left(2,b\right) = \{1\}$$
$$\delta'\left(\{1,2\},a\right) = \delta\left(\{1,2\},a\right) = \delta\left(1,a\right) \cup \delta\left(2,a\right) = \{1,2\} \cup \varnothing = \{1,2\}$$
$$\delta'\left(\{1,2\},b\right) = \delta\left(\{1,2\},b\right) = \delta\left(1,b\right) \cup \delta\left(2,b\right) = \{2\} \cup \{1\} = \{1,2\}$$

Also, $q'_0 = \{q_0\}$, where q_0 is the start state of the NFA; here, q'_0 = {1}. Further, final state set F' = {$R \in Q'$ | R contains an accept state of NFA}. We now have all information needed to draw up the equivalent DFA, as shown.



Problem 10.2: Let A = {p}, B = {p,q}, C = {p,r}, D = {p,q,r}, E = {p,q,s}, F = {p,q,r,s}, G = {p,r,s}, and H = {p,s}. Proceeding as we did in the previous problem, we write the following. Computations for states A to D are summarized below; the same reasoning applies to states E to H.

$$\delta'\left(A,0\right) = \delta\left(\{p\},0\right) = \delta\left(p,0\right) = \{p,q\} = B$$

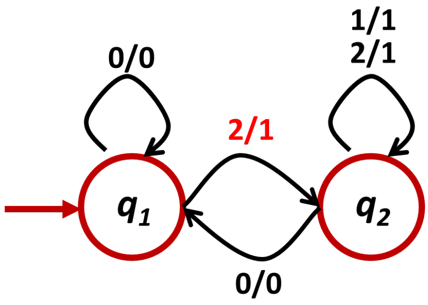
$$\delta'(A,1)=\delta(\{p\},1)=\delta(p,1)=\{p\}=A$$
$$\delta'(B,0)=\delta(\{p,q\},0)=\delta(p,0)\cup\delta(q,0)=\{p,q\}\cup\{r\}=\{p,q,r\}=D$$
$$\delta'(B,1)=\delta(\{p,q\},1)=\delta(p,1)\cup\delta(q,1)=\{p\}\cup\{r\}=\{p,r\}=C$$
$$\delta'(C,0)=\delta(\{p,r\},0)=\delta(p,0)\cup\delta(r,0)=\{p,q\}\cup\{s\}=\{p,q,s\}=E$$
$$\delta'(C,1)=\delta(\{p,r\},1)=\delta(p,1)\cup\delta(r,1)=\{p\}\cup\varnothing=\{p\}=A$$
$$\delta'(D,0)=\delta(\{p,q,r\},0)=\delta(p,0)\cup\delta(q,0)\cup\delta(r,0)=\{p,q\}\cup\{r\}\cup\{s\}=\{p,q,r,s\}=F$$
$$\delta'(D,1)=\delta(\{p,q,r\},1)=\delta(p,1)\cup\delta(q,1)\cup\delta(r,1)=\{p\}\cup\{r\}\cup\varnothing=\{p,r\}=C$$

The transition table is shown below.

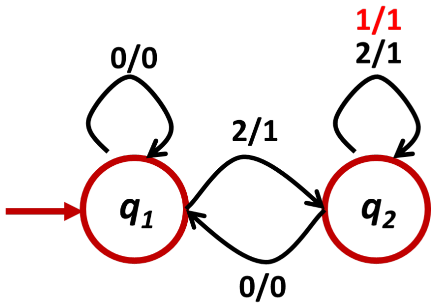
	0	1
→A	B	A
B	D	C
C	E	A
D	F	C
*E	F	G
*F	F	G
*G	E	H
*H	E	H

P.11 ➡ Solution

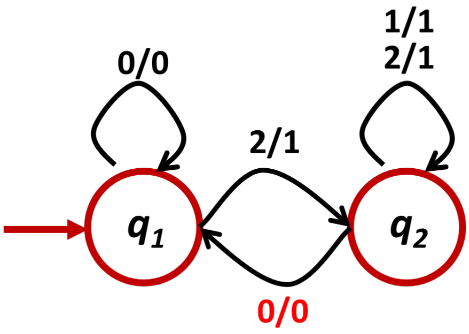
Problem 11.1: Machine T_1 begins at state q_1 and receives an input 2100. Upon receiving a 2, the machine transitions to q_2 and outputs a 1, as highlighted below.



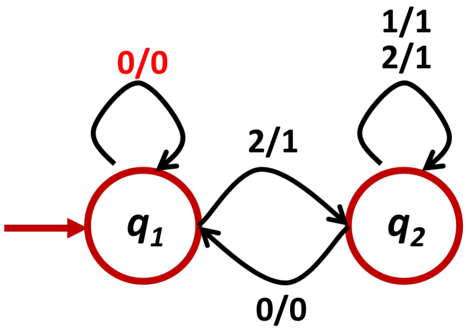
Upon receiving a 1, the machine remains at state q_2 and outputs a 1, as highlighted below.



Upon receiving a 0, the machine transitions to state q_2 and outputs a 0, as highlighted below.



Upon receiving a 0, the machine remains at state q_1 and outputs a 0, as highlighted on the next page.



Thus, the sequence of states entered is q_1, q_2, q_2, q_1, q_1. The output string is 1100.

Problem 11.2: Upon receiving an a, the machine transitions to state q_2 and outputs a 1. Upon receiving a b, the machine transitions to q_1 and outputs a 0. Upon receiving another b, the machine transitions to q_3 and outputs a 1. As the machine receives yet another b, it transitions to q_2 and outputs a 1. Upon receiving an a, the machine transitions to q_3 and outputs a 1. The sequence of states is $q_1, q_2, q_1, q_3, q_2, q_3$. The output string is 10111.

REFERENCES

- DU, D.-Z. and KO, K.-I. (2001). *Problem Solving in Automata, Languages, and Complexity*. Hoboken: John Wiley and Sons.
- HOPCROFT, J.E., MOTWANI, R. and ULLMAN, J.D. (2001). *Introduction to Automata Theory, Languages, and Computation*. 2nd edition. Upper Saddle River: Addison-Wesley.
- LINZ, P. (2017). *An Introduction to Formal Languages and Automata.* 6th edition. Burlington: Jones and Bartlett Learning.
- SIPSER, M. (2013). *Introduction to the Theory of Computation*. 3rd edition. Boston: Cengage Learning.