

Homework 2: Solutions

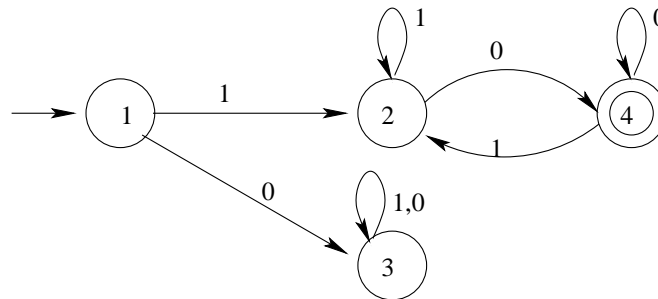
Due: 16 February 2005

Vinod Vaikuntanathan

Problem 1: Designing DFAs and NFAs. For each of the following, draw a state diagram for a DFA or NFA (as required) that recognizes the specified language. In all cases the alphabet is $\{0, 1\}$.

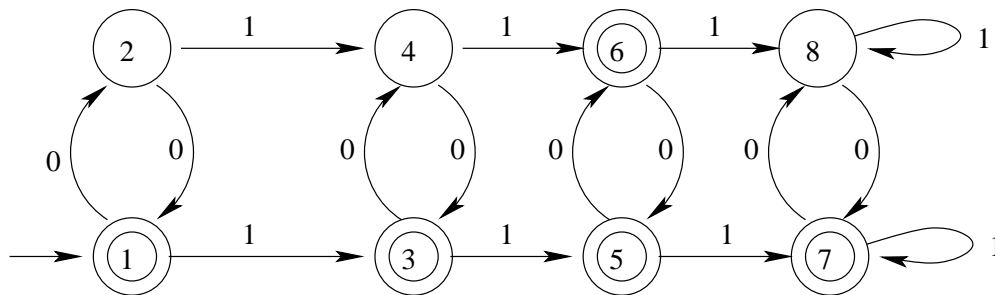
(a) (Sipser, Exercise 1.6, part a)

$L_a = \{w \mid w \text{ begins with 1 and ends with } 0\}$. Provide a DFA recognizing L_a .



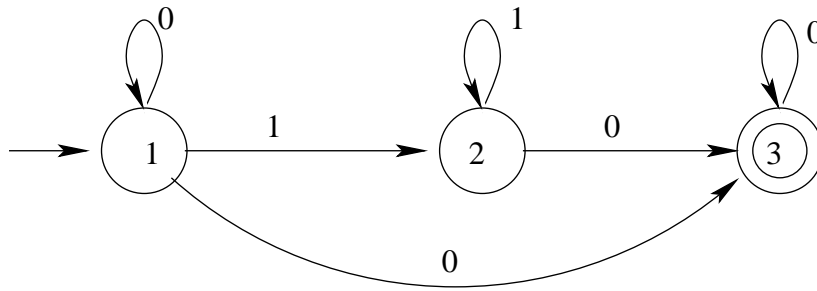
(b) (Sipser, Exercise 1.6, part 1)

$L_b = \{w \mid w \text{ contains an even number of } 0\text{s, or contains exactly two } 1\text{s}\}$. Provide a DFA recognizing L_b .

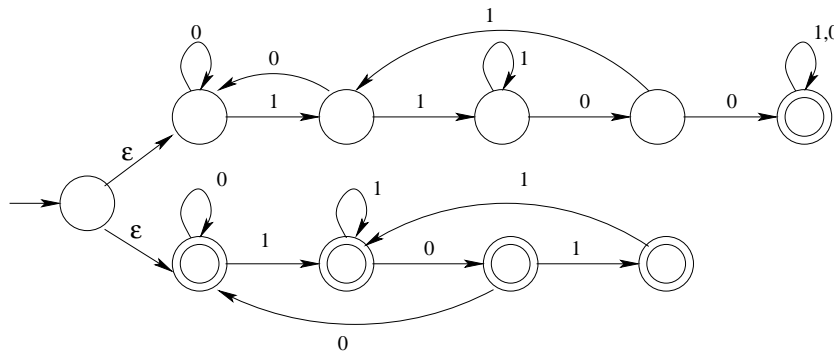


(c) (Sipser, Exercise 1.7, part e)

$L_c = 0^*1^*0^+$, that is, the set of strings consisting of some number (possibly zero) of 0s followed by some number of 1s followed by at least one 0. Provide an NFA recognizing L_c , with exactly three states.



(d) $L_d = \{w \mid w \text{ contains the substring } 1100 \text{ or does not contain the substring } 1010\}$. Provide an NFA recognizing L_d .



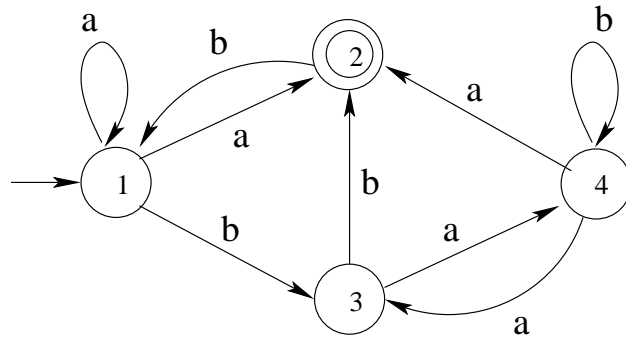
Problem 2: Proving an FA recognizes a language. For one of the automata you designed in problem 1, prove that the machine recognizes *exactly* the specified language. To do this, you will need to prove that your automaton (1) accepts all strings in the language and (2) does not accept any string not in the language.

Solution 2: (Note: A more complex example of proving that an FA recognizes a language is given in the recitation 2 solutions/discussion in the course page).

Any of the above languages can be chosen for this section; we demonstrate one way to prove that L_c is recognized by the NFA in solution 1.c. Any sound argument will be accepted.

- Suppose $x \in L_c$. Then x is of the form $1y0$ for some $y \in \Sigma^*$. $\delta^*(1, 1y0) = \delta^*(2, y0)$. Now, note that $\delta^*(2, y) \in \{2, 4\}$, since there is no path from $\{2, 4\}$ to states 1 or 3. Therefore, $\delta^*(2, y0) = \delta(\{2, 4\}, 0) = 4$, since from either state 2 or state 4, on reading a 0, we reach 4, the accept state.
- Suppose $x \notin L_c$. Then, either x does not begin with a 1 or x does not end with a 0. In the former case, on reading the first symbol, which is a 0, the machine ends up in a trap state, and therefore does not accept. In the latter case, again the machine does not accept, since all the transitions to the final state (i.e., state 4) are labeled by a 0.

Problem 3: NFA to DFA. Consider the following state diagram.



- (a) The state diagram above represents an NFA $N = (Q, \Sigma, \delta, q_0, F)$. Say what each of the components of the 5-tuple is, for this NFA.

Solution 3:

- (a) States $Q = \{1, 2, 3, 4\}$.
 (b) Alphabet $\Sigma = \{a, b\}$.

State	Input: a	Input: b
1	1,2	3
2	\emptyset	1
3	4	2
4	2,3	4

- (c) Transition Function $\delta : Q \times \Sigma \rightarrow Q$.

- (d) Start state $q_0 = 1$.
 (e) Accepting states $F = \{2\}$.

- (b) Apply the subset construction described in class to obtain a DFA $M = (Q', \Sigma, \delta', q'_0, F')$ that is equivalent to N . State the corresponding element(s) in Q', δ', q'_0, F' , then describe δ' via either a transition table or a state diagram.

Solution 3: As on page 55 of the text, we construct DFA $M = (Q', \Sigma, \delta', q'_0, F')$ that is equivalent to N .

- (a) $Q' = P(Q) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}, \{1, 2, 3\}, \{1, 2, 4\}, \{2, 3, 4\}, \{1, 3, 4\}, \{1, 2, 3, 4\}\}$. Our states in M will be the power-set of the states in N .
 (b) The alphabet $\Sigma = \{a, b\}$ remains the same.
 (c) The transition function δ' operates as follows. For $R \in Q'$ and $a \in \Sigma$ let $\delta'(R, a) = \{q \in Q \mid q \in \delta(r, a) \text{ for some } r \in R\}$.

State	Input: a	Input: b
\emptyset	\emptyset	\emptyset
$\{1\}$	$\{1,2\}$	$\{3\}$
$\{2\}$	\emptyset	$\{1\}$
$\{3\}$	$\{4\}$	$\{2\}$
$\{4\}$	$\{2,3\}$	$\{4\}$
$\{1,2\}$	$\{1,2\}$	$\{1,3\}$
$\{1,3\}$	$\{1,2,4\}$	$\{2,3\}$
$\{1,4\}$	$\{1,2,3\}$	$\{3,4\}$
$\{2,3\}$	$\{4\}$	$\{1,2\}$
$\{2,4\}$	$\{2,3\}$	$\{1,4\}$
$\{3,4\}$	$\{2,3,4\}$	$\{2,4\}$
$\{1,2,3\}$	$\{1,2,4\}$	$\{1,2,3\}$
$\{1,2,4\}$	$\{1,2,3\}$	$\{1,3,4\}$
$\{2,3,4\}$	$\{2,3,4\}$	$\{1,2,4\}$
$\{1,3,4\}$	$\{1,2,3,4\}$	$\{2,3,4\}$
$\{1,2,3,4\}$	$\{1,2,3,4\}$	$\{1,2,3,4\}$

- (d) $q'_0 = \{1\}$. Our start state in M is the element of $P(Q)$ corresponding to just the start state of N .
- (e) $F' = \{R \in Q' \mid R \text{ contains an accept state of } N\} = \{\{2\}, \{1, 2\}, \{3, 2\}, \{4, 2\}, \{1, 2, 4\}, \{1, 2, 3\}, \{2, 3, 4\}, \{1, 2, 3, 4\}\}$. Our accept states in M correspond to any elements of $P(Q)$ that contain an accept state from N .

Problem 4: Showing languages are regular. (From Sipser, Problems 1.31 and 1.32)

For any string $w = w_1w_2 \dots w_n$, the reverse of w , written w^R , is the string w in reverse order, $w_n \dots w_2w_1$. For any language A , let $A^R = \{w^R \mid w \in A\}$.

- Show that if A is a regular language, so is A^R .

Solution 4: Since A is a regular language, it is accepted by some DFA $D = (Q, \Sigma, \delta, q_0, F)$. We construct an NFA that recognizes A^R . Since the languages recognized by NFAs are exactly the regular languages, this suffices to prove our claim.

The NFA $N = (Q \cup \{q_{new}\}, \Sigma, \delta', q_{new}, F')$ is defined as follows: We retain the set of states of the DFA, and add a new state q_{new} . The alphabet remains the same. The new start state is the new state we added, q_{new} . $F' = \{q_0\}$. The transition function δ' is defined as follows:

$$\delta'(q_{new}, \epsilon) = F$$

$$\delta'(q_1, a) = \{q_2\} \quad \text{if and only if } \delta(q_2, a) = q_1 \quad \forall q_1, q_2 \in Q, a \in \Sigma$$

In words, we add a new state q_{new} , and add ϵ -transitions from q_{new} to all the final states of M (i.e., F). We also reverse all the arcs of the DFA M , and make $\{q_0\}$ (the set consisting of the start state of M) as the final state of N .

Claim 1 *The NFA N constructed above recognizes A^R if and only if M recognizes A .*

Proof: Let us see intuitively why this construction should work. A string w is in the language A if and only if there is a path (labeled with the symbols in w) that takes you from the start state q_0 to one of the final states in the DFA D . If (and only if) this is the case, we

are assured that there is a path from a state in F to q_0 in N , and therefore a path from q_{new} to q_0 in N .

Onto the formal proof: We want to prove by induction that for $q_1, q_2 \neq q_{new}$, $\delta^*(q_1, w^R) = q_2$ if and only if $\delta^*(q_2, w) = q_1$. Let us do this by induction on the length of w .

For the basis case, $\delta^*(q_1, \epsilon) = q_1$ (Remember that M is a DFA, and therefore, has no ϵ -transitions) and so is $\delta^*(q_1, \epsilon)$.

For the inductive step, assume that this is true for all strings of length at most k . Suppose we have a string wa of length $k + 1$ with $w \in \Sigma^*$ and $a \in \Sigma$. We know that $\delta^*(q_1, w) = q_2$ if and only if $\delta^*(q_2, w^R) = q_1$ for all $q_1, q_2 \neq q_{new}$. Suppose $\delta^*(q_1, wa) = q_2$ (for some q_1 and q_2 and some alphabet symbol a). This means, $\delta'(\delta^*(q_1, w), a) = q_2$. By definition of δ' , this means $\delta(q_2, a) = \delta^*(q_1, w)$. Now, by inductive hypothesis, this is true if and only if $\delta^*(\delta(q_2, a), w^R) = q_1$. This is equivalent to saying that $\delta^*(q_2, aw^R) = q_1$. (Note that aw^R is the reverse of wa).

Therefore, $\delta^*(q_0, w) = q_f$ for some $q_f \in F$ if and only if $\delta^*(q_f, w^R) = q_0$. This happens if and only if $\delta^*(q_{new}, w^R) = q_0$. Thus N accepts w if and only if D accepts w^R . ■

2. Let

$$\Sigma_3 = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \dots, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}$$

Σ_3 contains all size 3 columns of 0s and 1s. A string of symbols of Σ_3 gives three rows of 0s and 1s. Consider each row to be a binary number and let

$$B = \{w \in \Sigma_3^* \mid \text{the bottom row of } w \text{ is the sum of the top two rows}\}.$$

Show that B is regular. (Hint: Use part (a)).

Solution 4: First note that it is sufficient to prove that B^R is regular, by part (a). We proceed to give an automaton M that recognizes B^R . M has 2 states, q_0 , which intuitively denotes that the string that we have read so far leads to a carry 0, and q_1 that stands for carry 1. (Below, we make the convention that ϵ is in the language B .)

$M = (Q, \Sigma, \delta, q_0, F)$, where $Q = \{c_0, c_1\}$, Σ is the given alphabet, $q_0 = c_0$ and $F = \{c_0\}$. δ is specified as:

$$\delta(c_0, a) = c_0 \text{ if } a = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}.$$

$$\delta(c_0, a) = c_1 \text{ if } a = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

$$\delta(c_1, a) = c_1 \text{ if } a = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

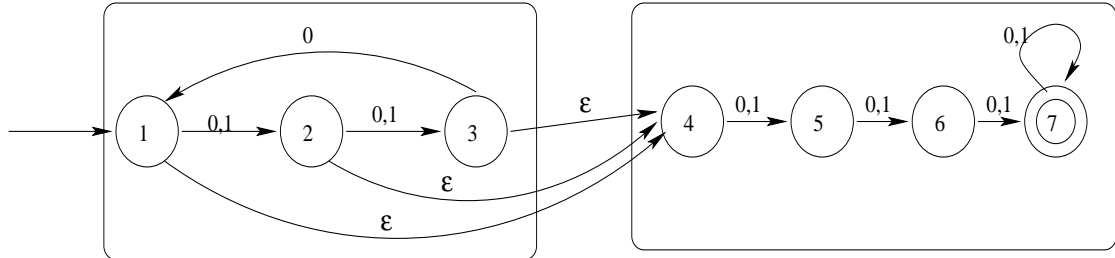
$$\delta(c_1, a) = c_0 \text{ if } a = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

All other arrows go to the trap state.

Problem 5:

- (a) **Closure construction.** Use the closure construction for concatenation described in class, and in Theorem 1.47, to show that L_1L_2 is regular, where L_1 is the set of strings containing 0 in every position that is a multiple of 3, and L_2 is the set of strings of length at least 3.

Solution 5: The NFA that describes the concatenation construction is as below:



- (b) **Regular languages are closed under another operation** (Sipser, Exercise 1.43)

Let A be any language. Define $DROPOUT(A)$ to be the language consisting of all strings that can be obtained by removing one symbol from a string in A . Thus, $DROPOUT(A) = \{xz|xyz \in A \text{ where } x, z \in \Sigma^*, y \in \Sigma\}$. Show that the class of regular languages is closed under the $DROPOUT$ operation. Give both a proof by picture and a more formal proof by construction as in Theorem 1.47.

Solution 5: We want to prove that if A is a regular language, so is $DROPOUT(A)$. Since A is regular, it is accepted by some DFA $M = (Q, \Sigma, \delta, q_0, F)$. We construct an NFA $N = (Q', \Sigma \cup \{\epsilon\}, \delta', q'_0, F')$ that will accept $DROPOUT(A)$.

Intuitively, there are 2 copies of the machine M , one of which corresponds to the state of having “not yet skipped a symbol” (copy 0) and the other corresponds to the state of having “already skipped a symbol” (copy 1). See the figure below.

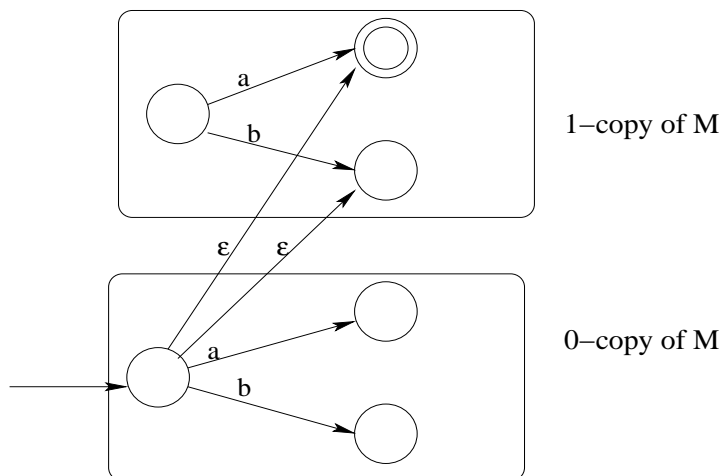
$Q' = \{(q, b) \mid q \in Q, b \in \{0, 1\}\}$. $q'_0 = (q_0, 0)$. The set of final states $F' = \{(q, 1) \mid q \in F\}$. δ' is defined as follows:

$$\delta'((q, b), a) = \{(\delta(q, a), b)\} \quad \text{for all } q \in Q, b \in \{0, 1\}, a \in \Sigma$$

What does this mean? Both the copy 0 and copy 1 of the machine do exactly as the original machine does on every symbol a of the alphabet.

$$\delta'((q, 0), \epsilon) = \{(\hat{q}, 1) \mid \exists a \in \Sigma, \delta(q, a) = \hat{q}\}$$

Also, at every stage, the machine has the option to skip a character. The only accepting states are in copy 1. This means, the machine cannot accept a string without skipping a character.



The formal proof is by induction on the length of the string. An appropriate inductive hypothesis is to assume that, for any string w of length k ,

- $\delta^*((q_0, 0), w) = (q_1, 0)$ if and only if $\delta^*(q_0, w) = q_1$. (Saying that the machine stays in the 0-copy if and only if it has not yet skipped a symbol) and
- $\delta^*((q_0, 0), w) = (q_1, 1)$ if and only if $\delta^*(q_0, w_1aw_2) = q_1$ for some $a \in \Sigma$ and $w = w_1w_2$. (Saying that the machine jumps to the 1-copy if and only if there is some symbol a that it skipped)

The details of induction, after this point, are straightforward and are omitted.