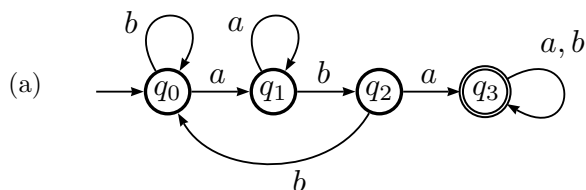
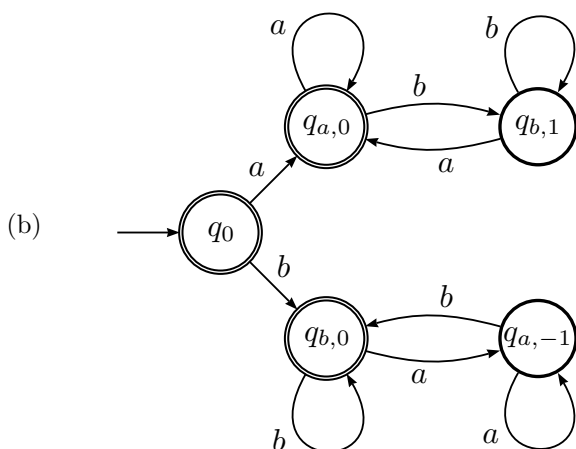


# CIS 262 Fall 2009: Homework 1 Solutions

## Problem 1



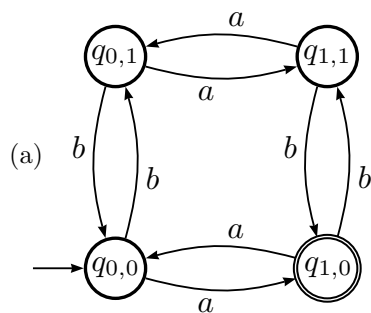
The state  $q_3$  is reached when the DFA has already seen  $aba$  as a subword somewhere in the string. The other three are reached when the last portion on the string that the DFA examined matched a prefix of  $aba$  and we are “waiting” for the remaining letters:  $q_0$  is reached when the longest matching prefix was  $\epsilon$ ,  $q_1$  when it was  $a$ , and  $q_2$  when it was  $ab$ .



The idea of the construction is that if we see  $ab$ , we must see  $ba$  before the next  $ab$  and vice-versa. So while the number of  $abs$  can be arbitrarily big, the *difference* between the number of  $abs$  and the number of  $bas$  can never be bigger than 1. This is why we only need a finite number of states.

In the DFA above,  $q_0$  is the initial state, where we have seen zero occurrences of both  $ab$  and  $ba$ , hence it should accept. The other states are labeled  $q_{x,n}$  where  $x$  is the last letter of the string so far, and  $n = (\text{number of } ab \text{ occurrences}) - (\text{number of } ba \text{ occurrences})$ . We make states accept when the difference between  $abs$  and  $bas$  is zero.

## Problem 2



(b) Write  $\#_a w$  for “the number of  $a$ s in the string  $w$ ”, and similarly  $\#_b w$  for the number of  $b$ s. We prove the following statement

For all  $w$ ,  $\hat{\delta}(q_{0,0}, w) = q_{i,j}$  iff  $i = \#_a w \pmod 2$  and  $j = \#_b w \pmod 2$ .

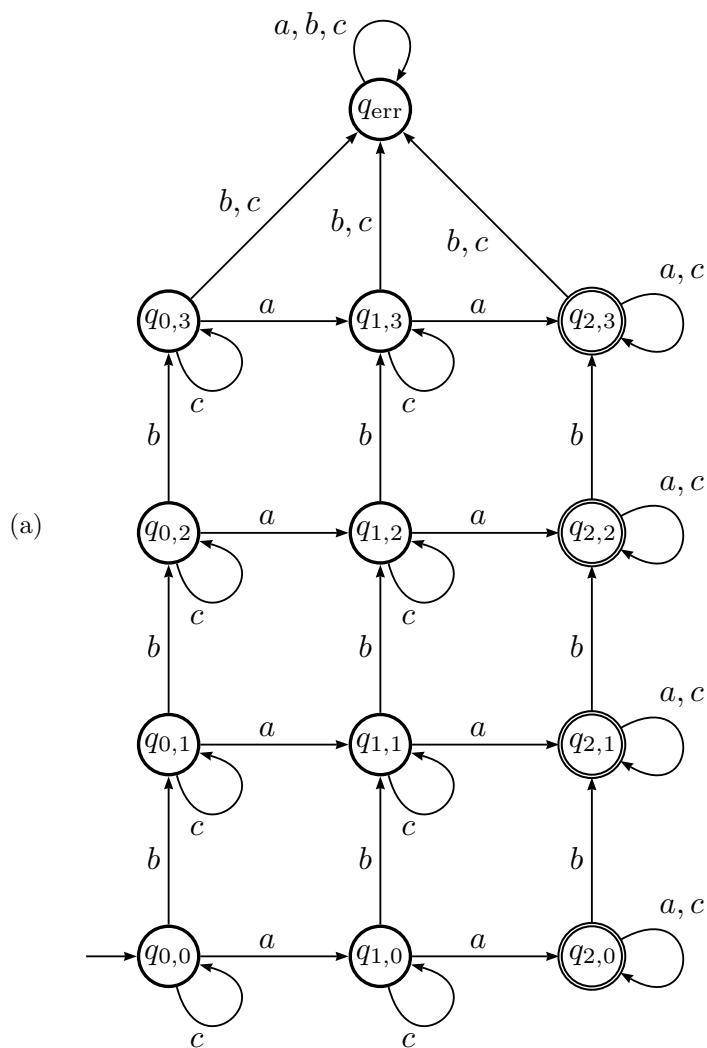
by induction on the structure of  $w$ . It then immediately follows that the DFA accepts the right language since  $F = \{q_{1,0}\}$ . The cases are:

- $w = \epsilon$ . Then  $\hat{\delta}(q_{0,0}, \epsilon) = q_{0,0}$  by definition of  $\hat{\delta}$ , while  $\#_a \epsilon \pmod 2 = \#_b \epsilon \pmod 2 = 0$  as required.
- $w = ux$  for some string  $u$  and some letter  $x$ . Write  $\hat{\delta}(q_{0,0}, u) = q_{i',j'}$ . Now,  $x$  is either  $a$  or  $b$  and  $q_{i',j'}$  is one of the four states, so there are eight cases to consider:
  - $x = a$  and  $q_{i',j'} = q_{0,0}$ . By the induction hypothesis,  $\#_a u \pmod 2 = i' = 0$  and  $\#_b u \pmod 2 = j' = 0$ , that is  $u$  contains an even number of  $a$ s and an even number of  $b$ s. So  $w = ua$  contains an odd number of  $a$ s and an even number of  $b$ s, and therefore we need to show that  $\hat{\delta}(q_{0,0}, w) = q_{1,0}$ . But

$$\begin{aligned}
 \hat{\delta}(q_{0,0}, w) &= \hat{\delta}(q_{0,0}, ua) \\
 &= \delta(\hat{\delta}(q_{0,0}, u), a) && \text{[by the definition of } \hat{\delta}.] \\
 &= \delta(q_{0,0}, a) && \text{[since we assumed } q_{i',j'} = q_{0,0}.] \\
 &= q_{1,0}
 \end{aligned}$$

- The other seven cases are similar.

### Problem 3



The general intuition is that states  $q_{i,j}$  represent strings with at least  $i$  as and at most  $j$  bs, while  $q_{\text{err}}$  represents strings with too many bs. More precisely  $q_{i,j}$  for  $i < 2$  represents strings with exactly  $i$  as and  $j$  bs;  $q_{2,j}$  represents strings with at least 2 as and exactly  $j$  bs; and  $q_{\text{err}}$  represents strings with more than 3 bs.

- (b) For general  $L_{n,m}$ , the automaton will have the same form as in part (a), only taller and wider. The states will still consist of one distinguished error state and  $(m+1)(n+1)$  states labeled with pairs of numbers. Now  $q_{i,j}$  for  $i < m$  represents strings with exactly  $i$  as and  $j$  bs;  $q_{m,j}$  represents strings with at least  $m$  as and exactly  $j$  bs; and  $q_{\text{err}}$  represents strings with more than  $n$  bs.

Formally, the automaton is  $(Q, \Sigma, \delta, q_0, F)$  where

- $Q = \{(i, j) \mid 0 \leq i \leq m \text{ and } 0 \leq j \leq n\} \cup \{\text{err}\}$
- $\Sigma = \{a, b\}$

- $\delta$  is defined by

$$\begin{aligned}
\delta((i, j), a) &= (i + 1, j) && \text{for } i < m \\
\delta((m, j), a) &= (m, j) \\
\delta((i, j), b) &= (i, j + 1) && \text{for } j < n \\
\delta((i, n), b) &= \text{err} \\
\delta((i, j), c) &= (i, j) \\
\delta(\text{err}, x) &= \text{err}
\end{aligned}$$

- $q_0 = (0, 0)$
- $F = \{(i, j) \in Q \mid i \geq m \text{ and } j \leq n\}$

(c) We claim that  $(m + 1)(n + 1) + 1$  is a precise lower bound. We need to prove two things: that it is a lower bound and that it is precise.

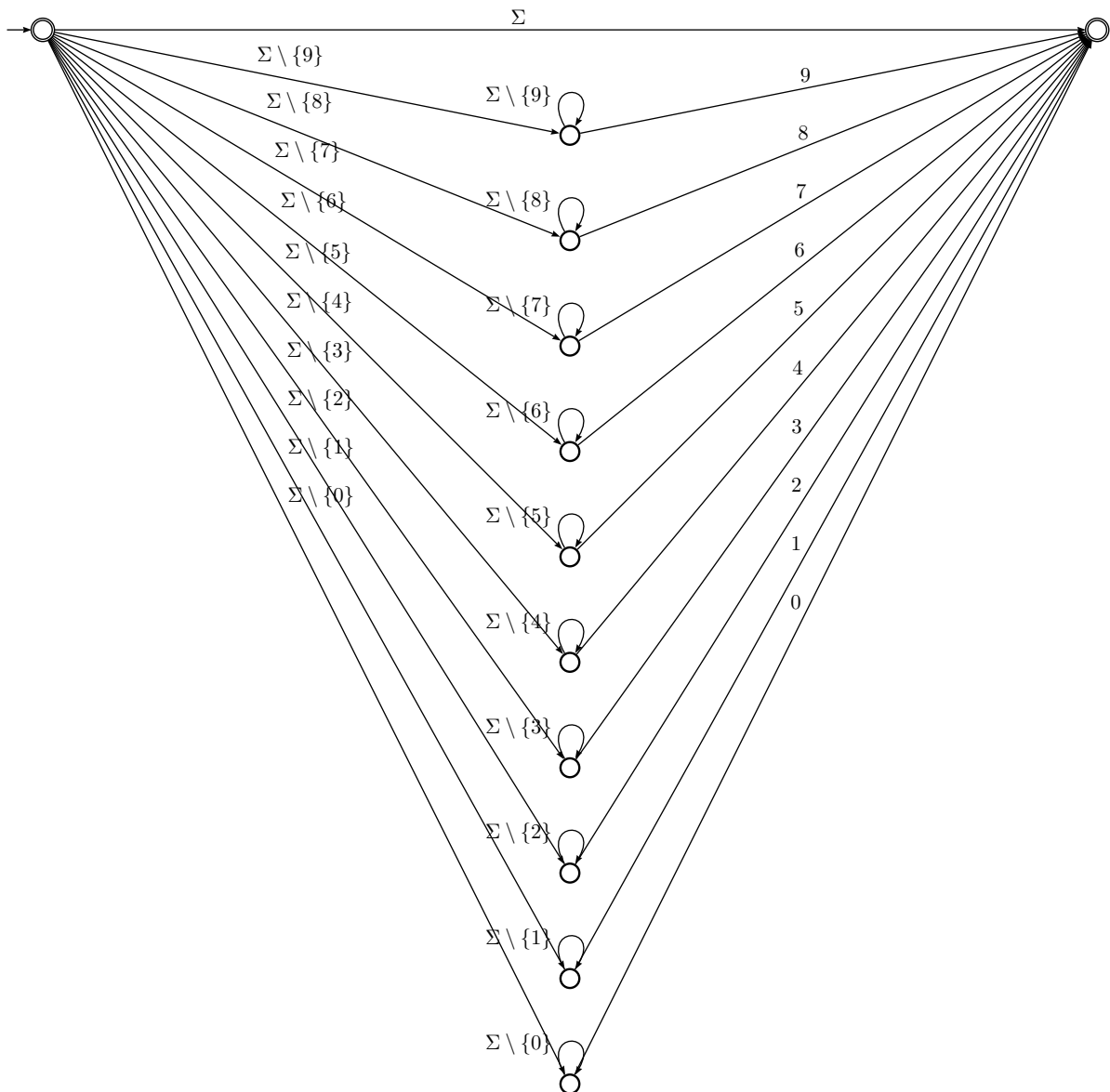
To show that it is a lower bound, we need to show that no DFA with fewer than  $(m + 1)(n + 1) + 1$  states can recognize  $L_{m,n}$ . So suppose  $D = (Q, \Sigma, \delta, q_0, F)$  is some DFA with fewer states; we need to show that it cannot accept the right language.

So consider the following set of  $(m + 1)(n + 1) + 1$  strings:  $A = \{a^i b^j \mid 0 \leq i \leq m, 0 \leq j \leq n\} \cup \{b^{n+1}\}$ . Since  $|A| > |Q|$ , by the pigeonhole principle there must exist two strings  $w, w' \in A$  such that  $\hat{\delta}(q_0, w) = \hat{\delta}(q_0, w')$ . That means that for all strings  $v$ ,  $D$  will either accept both  $wv$  and  $w'v$  or reject both  $wv$  and  $w'v$ . We show by a case analysis on  $w$  and  $w'$  that this means  $D$  is not accepting the right language:

- If  $w = b^{n+1}$  and  $w' = a^i b^j$ , then taking  $v = a^{m-i}$ ,  $D$  either accepts both  $wv$  and  $w'v$  or rejects them both. However,  $wv \notin L_{m,n}$  (it has too many  $b$ s) while  $w'v \in L_{m,n}$ .
- Otherwise,  $w = a^i b^j$  and  $w' = a^{i'} b^{j'}$  with  $i \neq i'$  or  $j \neq j'$ . There are four cases to consider:  $i < i'$ ,  $i > i'$ ,  $j < j'$  and  $j > j'$ . In the first case, suppose  $i < i'$ . Then taking  $v = a^{m-i'}$ ,  $wv \notin L_{m,n}$  (it has too few  $a$ s) while  $w'v \in L_{m,n}$ ; however  $D$  has to accept or reject both. The other three cases are similar.

To show that the bound is precise, we need to exhibit an DFA with  $(n + 1)(m + 1) + 1$  states which recognizes the right language. But this is exactly what we did in part (b).

### Problem 3



In the above diagram, some arrows are labeled with sets of digits as a shorthand for multiple arrows.

We claim that the above automaton is an NFA accepting  $L$ . That is,  $w \in L$  iff there exists a path in the automaton which is labeled  $w$  and leads to an accepting state. To see this, consider what paths there are to accepting states:

- The automaton can accept the empty string  $\epsilon$  by staying in the initial accepting state,  $\rightarrow \bigcirc$  .

• It can accept any single-digit string by taking a path along the uppermost arrow,  $\rightarrow \text{---} \text{---} \xrightarrow{\Sigma} \text{---} \text{---} \rightarrow$  .

• For any digit  $d \in \Sigma$ , it can accept a string of at least two digits where the last digit is  $d$  and has not

appeared before by taking a path through an intermediate state,

