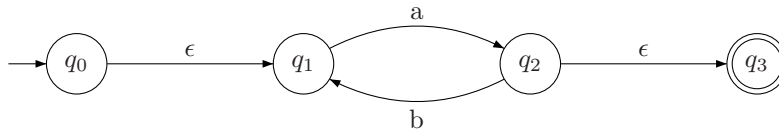


# CIS 262 Fall 2009: Midterm 1 Solutions

## Problem 1

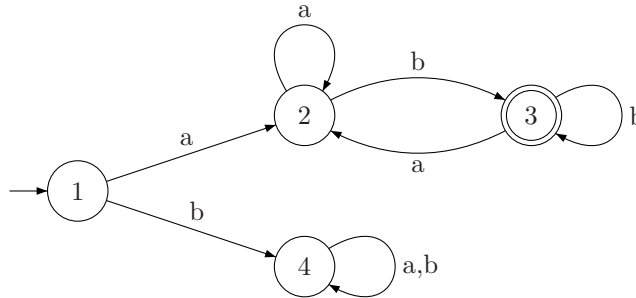
- (a) The language it accepts is trivial, i.e. either  $\emptyset$  or  $\Sigma^*$ . This is because all strings drive the automaton to the same state, so either all strings are rejected or all strings are accepted.
- (b) We can't conclude anything. For any language, it is possible to construct DFAs and NFAs with arbitrarily many states that recognize the language. So if  $k$  happens to be more than the minimal number of states to recognize  $L(A)$ , there will be automata  $B$  with both more and fewer states recognizing the same language.
- (c) Yes. The resulting expressions are guaranteed to be equivalent in the languages they generate, but they can be (syntactically) different. For instance, if we start with the automaton



then eliminating  $q_1$  first will result in  $a(ba)^*$  while eliminating  $q_2$  first results in  $(ab)^*a$ .

## Problem 2

- (a)



- (b)
  - $R_1$   $\epsilon$
  - $R_2$   $a + a(a + b)^*a$
  - $R_3$   $a(a + b)^*b$
  - $R_4$   $b(a + b)^*$

## Problem 3

Let  $L = \{0^n 1^m 2^{n+m} \mid n, m \geq 0\}$ . Suppose (to derive a contradiction) that  $L$  is regular, and let  $p$  be the pumping lemma constant for  $L$ . Consider the string  $w = 0^p 1^p 2^{2p}$ . Clearly  $w \in L$  and  $|w| \geq p$ , so the conditions for the pumping lemma are satisfied. Therefore  $w = xyz$  such that  $|xy| \leq p$  and  $|y| > 0$  and  $xy^kz$  for all  $k$ .

Since  $|xy| \leq p$  and the first  $p$  letters in  $w$  are all zeros, we know that  $x$  and  $y$  must both consist of all zeros. That means that  $xy^kz = 0^{(p-|y|+k|y|)} 1^p 2^{2p}$ . In particular, picking  $k = 0$  we should have  $xz = 0^{p-|y|} 1^p 2^{2p} \in L$ . But since  $|y| \neq 0$ ,  $(p - |y|) + p \neq 2p$ , so  $xz \notin L$ . This is a contradiction.

## Problem 4

In the following table, the names of the states indicate which states in the original NFA they correspond to. Final states are shown in bold (these are states corresponding to sets containing D). The initial state is A.

$q$	$\delta(q, a)$	$\delta(q, b)$
A	A	AB
AB	AC	ABC
AC	AD	ABD
<b>AD</b>	A	AB
ABC	ACD	ABCD
<b>ACD</b>	AD	ABD
<b>ABD</b>	AC	ABC
<b>ABCD</b>	ACD	ABCD

### Problem 5

All states in the automaton are reachable, so we can proceed to calculating the matrix of distinguishable states. We initialize the matrix to distinguish accepting from nonaccepting states:

	A	B	C
B	1		
C	1	0	
D	0	1	1

Next we start executing the loop of the algorithm. In the loop body, for each pair of states which is not already 1, we check for each letter in the alphabet where the two states go. If they go to a pair of states which is known to be distinguished, we update the matrix.

#### Iteration 1

- On reading  $a$ ,  $(C, B)$  goes to  $(B, B)$ . This does not tell us anything.
- On reading  $b$ ,  $(C, B)$  goes to  $(D, C)$ .  $D$  and  $C$  are distinguishable, so  $C$  and  $B$  must be too.
- On reading  $a$ ,  $(D, A)$  goes to  $(B, B)$ . This does not tell us anything.
- On reading  $b$ ,  $(D, A)$  goes to  $(D, A)$ . This does not tell us anything.

We update the matrix with the new information.

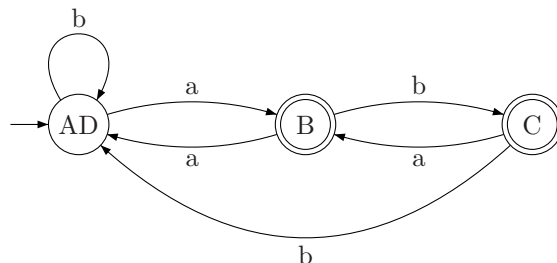
	A	B	C
B	1		
C	1	<b>1</b>	
D	0	1	1

#### Iteration 2

- On reading  $a$ ,  $(D, A)$  goes to  $(B, B)$ . This does not tell us anything.
- On reading  $b$ ,  $(D, A)$  goes to  $(D, A)$ . This does not tell us anything.

We have reached a stable solution and stop iterating.

The matrix tells us that the states  $A$  and  $D$  can be merged into one. Doing so we get the following DFA:



### Problem 6

(a)  $Insert(L) = (a + b)^*a(a + b)^*b(a + b)^*$

(b) Given a DFA  $A = (Q, \Sigma, q_0, \delta, F)$ , we define the NFA  $B = (Q, \Sigma, q_0, \delta', F)$  where

$$\delta'(q, x) = \{ \delta(q, x), q \} \quad \text{for all } q \in Q, x \in \Sigma$$

That is, the set of states, and the initial and accepting states are the same as in  $A$ , but we add extra arrows to each state that allow  $B$  to consume any letter while staying in the same state.

The idea is that if  $w \in L$  and  $w' \in Insert(w)$ , we can find an accepting run of  $B$  on  $w'$  by taking an accepting run of  $A$  on  $w$  and using the added arrows in  $B$  to consume the inserted letters. Conversely, given a  $w'$  accepted by  $B$  we can find an  $w \in L$  such that  $w' \in Insert(w)$  by deleting all letters that were consumed by one of the added arrows.