

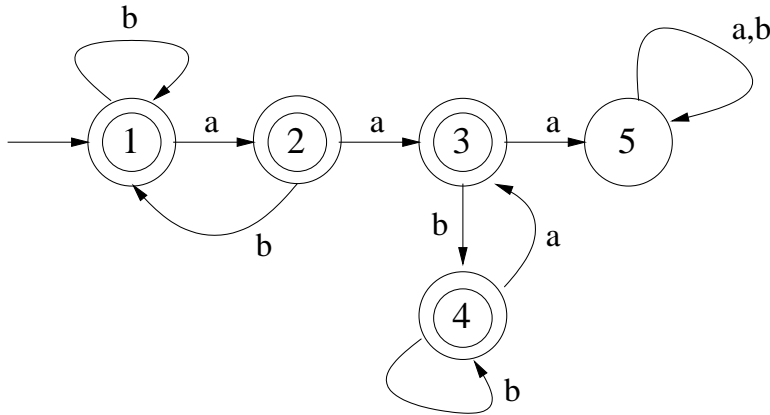
CS 273, Spring 2007

SOLUTIONS FOR Midterm Exam, Feb 22, 2007

Problem 1: DFA design (5 points)

Let L be the set of all strings over $\{a, b\}$ which contain at most one pair of consecutive a 's. For example, L contains $a, b, ab, aa, aab,$ and $abbbbaa;$ but not aaa or $aabaabba$. Construct a DFA that accepts L and give its state diagram. Explain briefly how your DFA works. You do not need to prove formally that your DFA is correct.

*You will receive **zero** credit if your DFA uses more than **10** states or makes significant use of non-determinism.*



A succinct DFA for the language is shown above. State 3 is reached for the first time when the first aa is seen. If one sees an a right away then we go to the reject state 5. Otherwise the machine loops between states 3 and 4 as long as two consecutive a 's are not seen. Many of you drew a separate branch from state 4 that would detect the next aa . That is easier to design but a little less succinct.

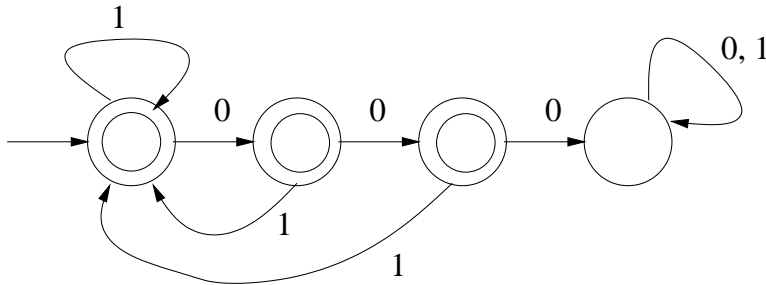
Problem 2: Regular Expressions (7 points)

No proof or explanation is required for this problem. But, if your answer is incorrect, sensible explanations may increase your partial credit.

- (a) (3 points) Give a DFA for the language defined by the regular expression $(1 + 01 + 001)^*(\epsilon + 0 + 00)$.

You will receive **zero** credit if your DFA uses more than 10 states or makes significant use of non-determinism

The first task is to identify the language of the regular expression which is $\{w \in \{0,1\}^* \mid w \text{ does not contain three consecutive 0's}\}$. A DFA for this is shown below.



Several of you tried to make DFA out of the regular expression thinking that there is an automatic procedure. There isn't one. One could convert the regular expression into an NFA and then convert it into a DFA but that is a painful process (some of you did it and got full credit).

- (b) (4 points) Give a regular expression for the language containing all strings of 0's and 1's such that every pair of adjacent 0's appears before any pair of adjacent 1's.

Another way to think about the language is as the set of strings $w \in \{0,1\}^*$ such that after the first 11 in w (from the left) there are no 00's. One can design the regular expression in two steps. First, let us consider those strings in the language that do not have a 11. A regular expression for such strings is $(0 + 10)^*(1 + \epsilon)$ where the term $(1 + \epsilon)$ allows strings that end in 1. Alternatively one can also write it as $(1 + \epsilon)(0 + 01)^*$. Second, let us consider those strings in the language that do have a 11. By considering the first 11 in the string, we realize that the part of the string before the 11 does not contain consecutive 1's and the part of the string after the 11 cannot contain consecutive 0's. Thus the regular expression for this is $(0 + 10)^*11(1 + 10)^*$. Thus a regular expression for the language is

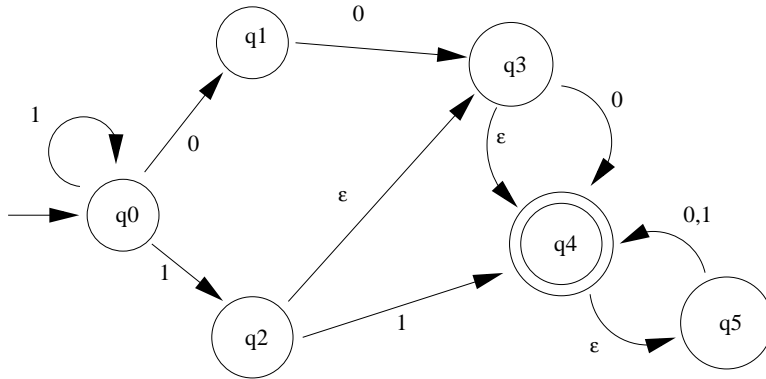
$$(0 + 10)^*(1 + \epsilon) + (0 + 10)^*11(1 + 10)^*.$$

However one can simplify the above into the following by noticing that 11 can be generated by $(1 + 10)^*$.

$$(0 + 10)^*(1 + 10)^*.$$

Problem 3: NFA transitions (7 points)

Suppose that the NFA $N = (Q, \Sigma, \delta, q_0, \{q_4\})$ is defined by the following state diagram:



Fill in the following values for the output of δ :

(a) $\delta(q_0, 1) = \{q_0, q_2\}$ (b) $\delta(q_4, 0) = \emptyset$

Recall that $E(S)$ is the epsilon closure of S . Fill in the values of the following expressions:

(c) $E(\{q_5\}) = \{q_5\}$

(d) $E(\{q_1, q_3\}) = \{q_1, q_3, q_4, q_5\}$

Suppose that we use the subset construction to construct a DFA M that recognizes the same language as N . Suppose that δ' is M 's transition function. Fill in the following values for the output of δ' :

(e) $\delta'(\{q_0, q_2\}, 1) = \{q_0, q_2, q_3, q_4, q_5\}$

(f) $\delta'(\{q_1\}, 1) = \emptyset$

(g) What is M 's start state? $\{q_0\}$

Answers for the 24 Feb makeup were: (a) \emptyset , (b) $\{q_4\}$, (c) $\{q_2, q_3, q_4, q_5\}$, (d) $\{q_0, q_4, q_5\}$, (e) $\{q_4, q_5\}$, (f) $\{q_1, q_4, q_5\}$, (g) $\{q_0\}$.

This question looks very simple, but it's easy to get details wrong. A lot of people got more-or-less half credit.

Notice that parts (a) and (b) are asking for the original one-step transition function for the NFA. This requires following only the 0 or 1 arcs from the input state. You should not follow any epsilon transitions.

In parts (e) and (f), computing the right answer requires following the 0 or 1 transitions from the input states, then taking the epsilon closure, **in that order**.

Writing $\{\}$ in place of \emptyset is incorrect formal math style, but no points were deducted for it this time.

Half-credit was awarded for missing or extra set brackets, most examples of one state missing from a list, not including the input states in (c) and (d), doing the two steps for (e) and (f) in the wrong order (epsilon transitions and then 0 or 1 arcs), and writing “sink state” as an answer to (f) without identifying it as the empty set.

0 points were given for parts that had more than one of the half-point errors, involved following epsilon transitions in parts (a) and (b), and involved listing states other than q_0 in part (g).

Problem 4: NFA modification and tuple notation (8 points)

Given an NFA M that accepts the language L , design a new NFA M' that accepts the language $L - \{\epsilon\}$. Use tuple notation assuming that $M = (Q, \Sigma, \delta, q_0, F)$. Note that the input NFA M may have ϵ -transitions. You may use the notation in the book $E(A)$ for the ϵ -closure of A where A is a set of states. Be careful about notation; both input and output are NFAs.

(a) Briefly explain the idea behind your construction, using English and/or pictures.

A NFA accepts ϵ if and only if $E(\{q_0\})$ contains a final state. Thus, to prevent M from accepting ϵ one naive option is to mark each final state in $E(\{q_0\})$ as a non-final state but this might disallow other strings in L from being accepted. In general it is hard to know what M 's behaviour is. The other option is simulate M 's behaviour on w for the first character of w (from Σ) and then let M take over. This can be done by adding a new start state q'_0 and adding transitions from q'_0 to the states of M on each character $a \in \Sigma$. We need to ensure that all states that M can reach on a are reached by the new machine on a . The states reachable by M on the first character a are given by $\cup_{q \in E(\{q_0\})} \delta(q, a)$ (note that one can take the ϵ -closure of this but that is not necessary). Thus q'_0 on a should reach all those states. Note that q'_0 does not have any transitions on ϵ and has no incoming transitions and thus once the first character is seen the new machine behaves exactly like the old machine.

(b) Give the details of your construction, using tuple notation.

$$M' = (Q', \Sigma, \delta', q'_0, F) \text{ where } Q' = Q \cup \{q'_0\}.$$

The transition function δ' is given below.

$$\delta'(q'_0, a) = \cup_{q \in E(\{q_0\})} \delta(q, a) \quad a \in \Sigma$$

$$\delta'(q, a) = \delta(q, a) \quad q \in Q, a \in \Sigma$$

Alternate Solutions: Some of you gave alternate solutions for full credit though they were not as short as the one above.

Some solutions involved more complicated variants of the construction above. One involved making two copies of M and starting with one copy in which there are no final states and then moving into the original copy of M with final states after reading the first character of the input string. A variant of this involved duplicating only the states in $E(\{q_0\})$.

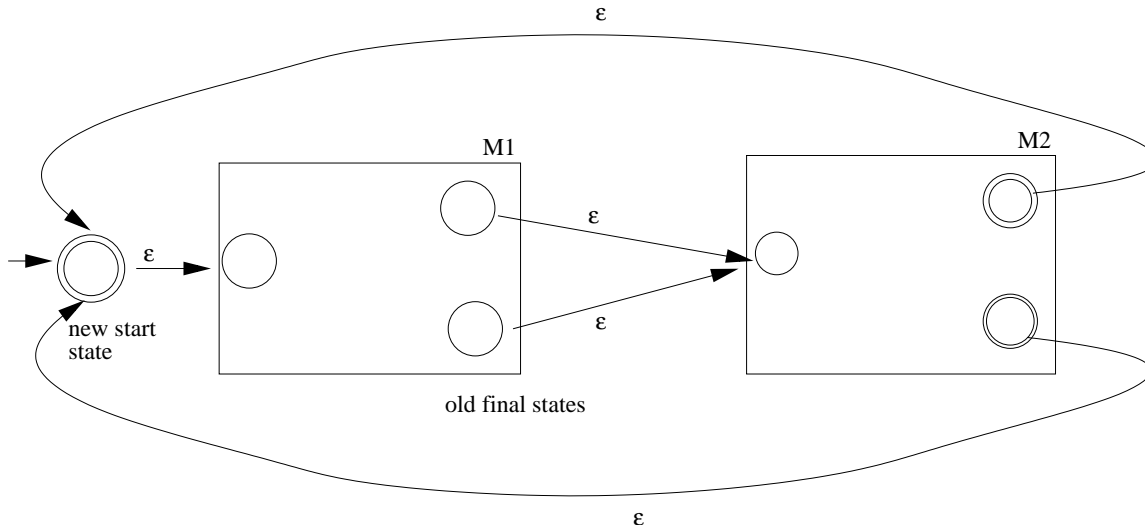
One interesting solution is to observe that $L - \{\epsilon\}$ is the same as $L \cap L'$ where $L' = \Sigma^* - \{\epsilon\}$. That is, L' is the set of *all* strings of length 1 or more. There is a simple DFA for L' with two states. Then $L \cap L'$ can be done by product construction for intersection if one has a DFA for L . To get a DFA for L one can use the subset construction to convert the NFA M into a DFA. This results in a long description. We haven't discussed this in class but a product construction can be done directly for the intersection of two languages given by NFA's instead of DFA's which would have simplified the description considerably.

2 pts were given for realizing that M can accept ϵ even if q_0 was not a final state - no further points were given if the construction simply omitted these final states or removed the states from M . 4-5 points were given for a construction that worked if M is a DFA instead of a NFA.

Problem 5: Short explanation (8 points)

The answers to these problems should be short and not complicated.

(a) Suppose that L_1 and L_2 are regular languages, recognized by NFAs M_1 and M_2 . Give a box diagram which shows how to construct an NFA recognizing the language $(L_1 \cdot L_2)^*$.



It was important that you showed a start state and multiple final states for both M_1 and M_2 , labelled your arcs (i.e. with epsilons), indicated which states of the new machine were final, and included a new start state. A couple people avoided point deductions by including comments that disambiguated their diagrams, e.g. made clear that they intended multiple final states even though they had drawn only one.

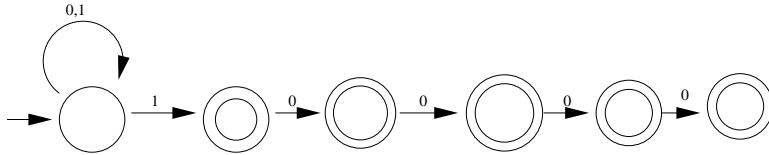
The old final states of M_1 should ideally be shown as non-final in the box diagram, with some sort of comment on them that they used to be final. However, I wasn't picky about taking off points for this.

Some variations are possible. E.g. you can send the loop arcs into the start state of M_1 rather than into the new initial state. Or, if they go into the new final state, you can omit the final markings on the old final states of M_2 .

The solution for the makeup exam is similar, but the two automata are connected in parallel and the loop only goes around one of them.

(b) Give a NFA for the language $L = \{w \in \{0,1\}^* \mid \text{there is a 1 in the last 5 positions of } w\}$. For example, 0000100 and 00010 are in L but 1100000 is not. Note that a string of length less than or equal to 5 would be in L if and only if it contains a 1. A DFA for this will be too large. *You will receive **zero** credit if your NFA uses more than **10** states.*

Here is the most succinct solution:



There were several other reasonably-succinct solutions, including a DFA. (Yup: the comment about DFAs on the problem was wrong. We're sorry.) Points were deducted for automata which worked but were much too complex, typically involving 10 states and lots of transitions.

Problem 6: Product construction (6 points)

Suppose that we are given DFAs M_A , M_B , and M_C recognizing languages L_A , L_B , and L_C . We can use the product construction to build a new DFA M recognizing the language $(L_A \cup L_B) - L_C$.

Specifically, suppose that

$$M_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$$

$$M_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$$

$$M_C = (Q_C, \Sigma, \delta_C, q_C, F_C)$$

Using good mathematical notation, specify the following components of M in terms of the corresponding components of M_A , M_B , and M_C . You can assume that $M = (Q, \Sigma, \delta, q_0, F)$.

(a) The set of states in M

$$Q_A \times Q_B \times Q_C.$$

(b) The set of final states of M .

$$\{(r, s, t) \in Q_A \times Q_B \times Q_C \mid (r \in F_A \text{ or } s \in F_B) \text{ and } t \notin F_C\}$$

(c) M 's transition function δ .

$$\delta((r, s, t), a) = (\delta(r, a), \delta(s, a), \delta(t, a))$$

The solution for the 24 Feb makeup differs only in details of the logic operations in (b).

Problem 7: True/false (9 points)

Label each of the following claims as true or false. (No explanation is required.)

- (a) Every language that is not regular is infinite. TRUE: every finite language is regular
- (b) Given regular expressions R_1 and R_2 , we can produce another regular expression S such that $L(S) = \overline{L(R_1)} - L(R_2)$. TRUE: closure properties
- (c) An NFA N that accepts the string ϵ has to have the start state as an accepting state. FALSE: start state might have epsilon transition to an accepting state
- (d) Any regular language can be accepted by an NFA with exactly one accept state. TRUE: done in class, create a new accept state with epsilon transitions to it
- (e) \emptyset^* is the same language as \emptyset . FALSE: The star of a set always contains the empty string
- (f) There is a bijection between any two infinite regular languages. TRUE: sets of strings are always countable and there is a bijection between any two countable infinite sets.
- (g) There is a language L that is recognized by a DFA M with n states but \overline{L} is not recognized by any DFA with $\leq 2^n$ states. FALSE: swap the accept and non-accept markings in M
- (h) Let r and s be regular expressions, then $L((r^*s^*)^*) = L((r + s)^*)$. TRUE: both contain any string of r 's and s 's
- (i) When converting a NFA N to a DFA M using the subset construction the state in M corresponding to \emptyset can have a transition going to another state than itself. FALSE: this is the sink state for the DFA

The answers for the 24 Feb makeup were: T, T, F, F, F, T, F, F, T