

CS 273
Exam 1
Fall 2006
October 3, 2006

INSTRUCTIONS (read carefully)

- Fill in the following information giving name and ID.

NAME:

ID:

- CLEARLY print your name and ID on every page.
- There are 6 problems, on pages numbered 1 through 7. Make sure you have a complete exam.
- The point value of each problem is indicated next to the problem, and in the table below.
- Points may be deducted for solutions which are correct but excessively complicated or poorly explained.
- The exam is designed for slightly over one hour, although you have two hours.
- It is probably wise to glance at all problems and point values before beginning, to best plan your time.
- This is a closed book exam. No notes of any kind are allowed. Do all work in the space provided, using the backs of sheets if necessary. See the proctor if you need more paper.

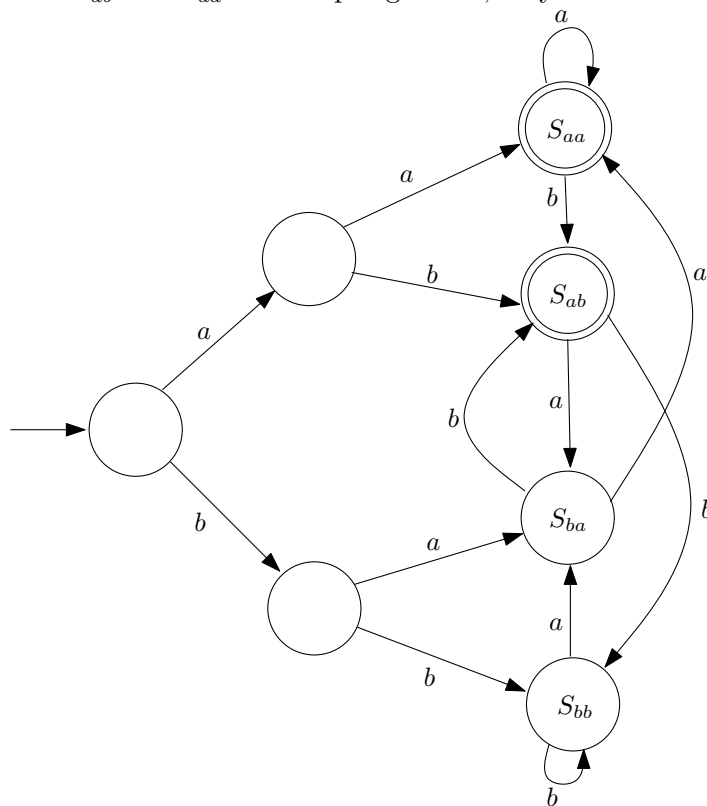
Problem	Possible	Score
1	6	
2	8	
3	6	
4	10	
5	12	
6	8	
Total	50	

Problem 1: DFA design (6 points)

Let L be the set of all strings over $\{a, b\}$ in which the second symbol from the right end is a . For example, L contains aa , aab , and $abbbbbaa$; but not ϵ or $aaba$. 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.*

Solution Basically we need to accept exactly those strings which end in ab or aa . By checking the transitions of the following DFA, one can verify that a given string ending in xy will end in state S_{xy} , and since only states S_{ab} and S_{aa} are accepting states, only desired strings will be accepted.



Main source of error: Most people who made mistakes have designed an automata that accepts a language which is just a subset/superset of the desired language.

Problem 2: Regular Expressions (8 points)

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

- (a) Let $L = \{0^n 1^m \mid m \geq 1 \text{ and } n + m \text{ is even}\}$. Give a regular expression for the language L .
*You will receive **zero** credit if your regular expression is more than **35** characters long.*

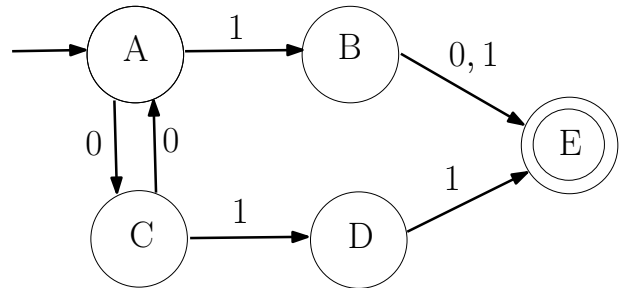
Solution: A regular expression for L is $(00)^*(11)^+ + (00)^*01(11)^*$.

Since $n + m$ is even and $m \geq 1$, we have two cases: (i) Both n and m are even, and $m \geq 2$. This case can be described by $(00)^*(11)^+$. (ii) Both n and m are odd. This case can be described by $(00)^*01(11)^*$.

- (b) Give a DFA for the language defined by the regular expression $(0^*11) + ((00)^*10)$. You can omit the “dead” state and the transitions into it, but your automaton must be **deterministic**.
*You will receive **zero** credit if your DFA uses more than **10** states (excluding the “dead” state) or if it makes significant use of non-determinism.*

Solution: We can write the regular expression $(0^*11) + ((00)^*10)$ as $(00)^*1(1+0) + 0(00)^*11$. Thus, we construct the required DFA as follows.

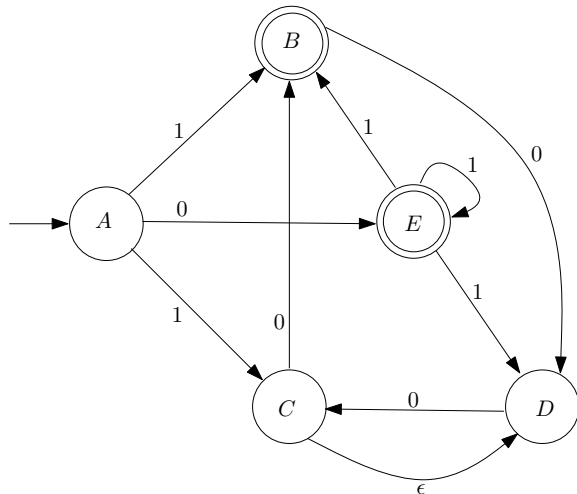
Intuitively, **A** represents the state that has seen an even number of 0's, **B** represents the state that has seen an even number of 0's followed by a 1, **C** represents the state that has seen an odd number of 0's, **D** represents the state that has seen an odd number of 0's followed by a 1, and **E** represents the accept state.



Problem 3: NFA to DFA conversion (6 points)

Convert the following NFA to a DFA recognizing the same language. An explanation is not required for full credit. However, showing your work and putting informative labels on your states may increase your partial credit if you make a mistake. Hint: remove the epsilon transition first.

Solution:



$$\delta(A, 0) = \{E\}$$

$$\delta(A, 1) = \{B, C, D\}$$

$$\delta(B, 0) = \{D\}$$

$$\delta(B, 1) = \{\}$$

$$\delta(C, 0) = \{B, C, D\}$$

$$\delta(C, 1) = \{\}$$

$$\delta(D, 0) = \{C, D\}$$

$$\delta(D, 1) = \{\}$$

$$\delta(E, 0) = \{\}$$

$$\delta(E, 1) = \{B, D, E\}$$

so

$$\delta(B, 0) \cup \delta(C, 0) \cup \delta(D, 0) = \{B, C, D\}$$

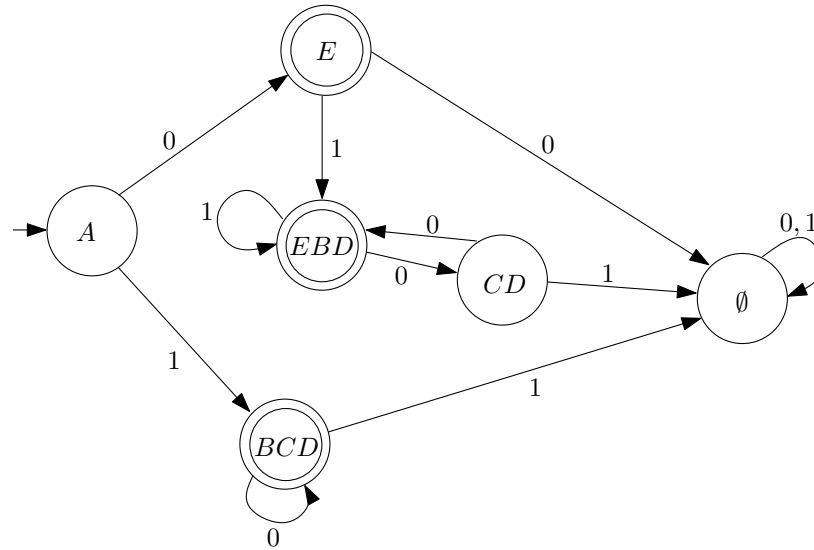
$$\delta(B, 1) \cup \delta(C, 1) \cup \delta(D, 1) = \{\}$$

$$\delta(B, 0) \cup \delta(D, 0) \cup \delta(E, 0) = \{C, D\}$$

$$\delta(B, 1) \cup \delta(D, 1) \cup \delta(E, 1) = \{B, D, E\}$$

$$\delta(C, 0) \cup \delta(D, 0) = \{B, C, D\}$$

$$\delta(C, 0) \cup \delta(D, 0) = \{\}$$



Problem 4: DFA modification and tuple notation (10 points)

Let $\Sigma = \{a, b, c, d\}$.

If w is a string in Σ^* , let's define $\text{subdivide}(w)$ to be the new string created by adding a hash sign ($\#$) after every character in w . That is, if $w = a_1a_2\dots a_n$ (where each a_i is a character in Σ) then $\text{subdivide}(w) = a_1\#a_2\#\dots a_n\#$. For example, if w is the string *cacca*, then $\text{subdivide}(w)$ is the string *c#a#c#c#a#*.

Given a language $L \subseteq \Sigma^*$, define the language $\text{subdivide}(L)$ to be $\{x \mid \exists w \in L \text{ such that } x = \text{subdivide}(w)\}$. Notice that the alphabet for $\text{subdivide}(L)$ is $\Sigma \cup \{\#\}$

Prove that if L is regular, then $\text{subdivide}(L)$ is also regular. Hint: L is recognized by some DFA. Explain how to modify this DFA into a new DFA or NFA recognizing $\text{subdivide}(L)$. Briefly describe the idea behind your construction and give the details in tuple notation.

Solution: Because L is regular, it is recognized by some DFA $M = (Q, \Sigma, \delta, q_0, F)$. To create a new DFA N which recognizes $\text{subdivide}(L)$, you need to insert a new state in the middle of each of M 's transitions. There's three ways to label the new states and work out the details.

1. For each old state q_i , create a new state r_i . Each transition that originally went to q_i is now redirected to r_i . And you add a new transition from r_i to q_i on input $\#$.
2. For each old state q_i , create a new state r_i . Each transition out of q_i is removed and replaced by a similar transition from r_i . And you add a new transition from q_i to r_i on input $\#$. You must also modify the start state to be r_0 and the accept states to be $\{r_i \mid q_i \in F\}$, otherwise the $\#$'s will end up before each original character rather than after it.
3. For each transition from q_i to q_j on symbol a , you add a new state $r_{i,a}$. The original transition is then replaced by a transition from q_i to $r_{i,a}$ on symbol a , and then a transition from $r_{i,a}$ to q_j on input $\#$.

Most of the points were for getting this far. Some common bugs in the basic structure of the construction:

- Very serious: Creating a single new state $q_\#$. The old states transition into $q_\#$. Magic is then used to get into the correct state after $q_\#$.
- Somewhat serious: like solution 3, except that the new states are named r_i . The new states aren't remembering enough to know which state to go to after they read the $\#$.
- Minor: Not modifying the start or accept states for solution 2.
- Minor: For solution 3, you are creating one new state per transition. Many people who used solution 3 lost a couple points for claiming they needed one new state for each old state.

Most full or nearly full credit solutions produced a clear description using a combination of English, fragments of state diagrams, and tuple notation. However, I didn't require any specific mix: one person got full credit for a particularly clear English description of the construction. Many people lost a point or two for descriptions that were vague or hard to understand or misused tuple notation.

A small number of people ignored the hint and proved that $\text{subdivide}(L)$ is regular by exhibiting a homomorphism from L to $\text{subdivide}(L)$. That wasn't what I intended but it answered the question as written, so completely correct versions were worth full credit.

Problem 5: Short explanation (12 points)

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

- (a) A DFA $M = (Q, \Sigma, \delta, q_0, F)$ (δ is a function from $Q \times \Sigma$ to Q) accepts a string w if $w = w_1w_2\dots w_n$ (where each w_i is a character in Σ) and there is a sequence of states $r_0r_1\dots r_n$ (each r_i is a member of Q) such that

1. $r_0 = q_0$
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, \dots, n - 1$, and
3. $r_n \in F$

How must this definition be modified if M is an NFA?

Solution: An NFA $M = (Q, \Sigma, \delta, q_0, F)$ (δ is a function from $Q \times \Sigma \cup \{\epsilon\}$ to the powerset of Q) accepts a string w if $w = w_1w_2\dots w_n$ (where each w_i is a character in $\Sigma \cup \{\epsilon\}$) and there is a sequence of states $r_0r_1\dots r_n$ (each r_i is a member of Q) such that

1. $r_0 = q_0$
2. $r_{i+1} \in \delta(r_i, w_{i+1})$, for $i \in \{0, 1, 2, \dots, n - 1\}$ and
3. $r_n \in F$

2 points were awarded for realizing that $\delta : Q \times \Sigma \cup \{\epsilon\} \rightarrow P(Q)$ and 2 points were awarded for changing the description of which strings are accepted.

- (b) Assume we know that the language $L = \{0^n1^n \mid n \geq 0\}$ is not regular. Define the language $L' \subseteq \{a, b, c, \#\}^*$ to be $\{a^n\#(bc)^n \mid n \geq 0\}$. Use closure properties to prove that L' is not regular.

Solution: See other file.

- (c) The language $L \subseteq \{a, b\}^*$ is defined by the following recursive rules:

1. ϵ is in L .
2. If x and y are strings in L , then $axby$ and $bxay$ are also in L .
3. L is the smallest set satisfying conditions (1) and (2).

Give a non-recursive definition for L and explain briefly why it defines the same language as the recursive definition. (A formal proof is *not* required and is, in fact, a bit challenging to come up with.)

Solution: See other file.

Problem 6: True/false (8 points)

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

- (a) Given any regular expression R , we can produce another regular expression S such that $L(S) = \overline{L(R)}$.

True. This is by closure property of regular expressions.

- (b) If L is regular and L' is not regular, then $L \cup L'$ is not regular.

False. Take a non-regular language L over $\{0,1\}^*$; let $L' = \{0,1\}^* \setminus L$. Note that L' is also non-regular (because otherwise, L would be regular, by the closure property). Also note that $L \cup L' = \{0,1\}^*$, which is regular.

- (c) Let $\Sigma = \{a,b,c\}$. Define the relation \sim on Σ^* such that, for any strings w and y in Σ^* , $w \sim y$ if and only if w and y contain the same number of a 's or w and y contain the same number of b 's. For example $aaba \sim bbabaa$ and $bccb \sim abab$ but $bbcc \not\sim bbbacc$. The relation \sim is an equivalence relation.

False. It is easy to verify that $aaba \sim aab$ and $aab \sim aabb$. If the relation \sim is an equivalence relation, then we have $aaba \sim aabb$, which is not true by the definition of \sim .

- (d) Any regular language can be accepted by a DFA with exactly one accept state.

False. See solution to HW3 problem 3B.

- (e) Let Σ be the set of ASCII alphabetic characters (i.e. the lowercase and uppercase alphabets). Define L by

$$L = \{w\#w \mid w \in \Sigma^* \text{ and } w \text{ contains no duplicated characters}\}$$

For example, $abc\#abc$ is in L but $bcc\#bcc$ is not in L . L is regular.

True. There are only a finite number of strings in L . Therefore, L is regular.

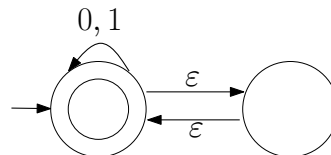
- (f) A language L satisfies the conditions of the pumping lemma if and only if L is regular.

False. This is mentioned in the lecture on Sep. 26. See http://en.wikipedia.org/wiki/Pumping_lemma_for_regular_languages for more information.

- (g) Suppose that M is an NFA. Let M' be an NFA just like M , except that the accept markings have been swapped. (That is, the accept states have been relabelled non-accept and the non-accept states have been relabelled accept.) $L(M') = \overline{L(M)}$

False. Suppose M is defined by the state diagram on the right.

It is easy to verify that $L(M') = L(M) = \{0,1\}^*$.



- (h) Let L be the language defined by the regular expression a^* . Let L' be the language defined by the regular expression b^* . L and L' are disjoint (i.e. have an empty intersection).

False. The empty string is in both sets.