

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.*

Problem 2: Regular Expressions (8 points)

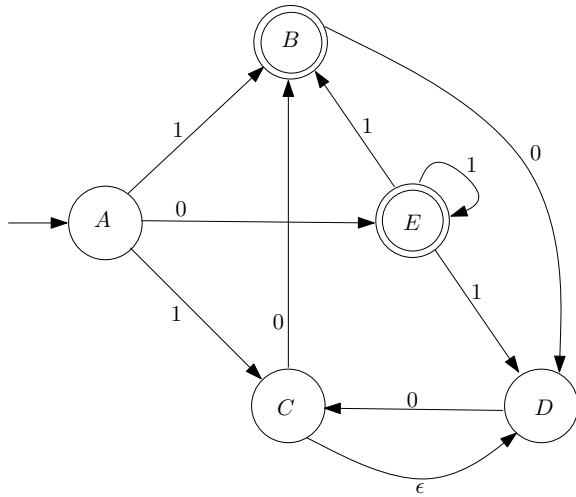
No proof or explanation is required for this problem. But, if your answer is incorrect, sensible explanations may increase your partial credit.

- (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.*

- (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.*

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.



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.

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?

- (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.

(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 axy and bxy 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.)

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 $L(S) = \overline{L(R)}$.
- (b) If L is regular and L' is not regular, then $L \cup L'$ is not 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.
- (d) Any regular language can be accepted by a DFA with exactly one accept state.
- (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.

- (f) A language L satisfies the conditions of the pumping lemma if and only if L is regular.
- (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.) $L(M') = \overline{L(M)}$
- (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).