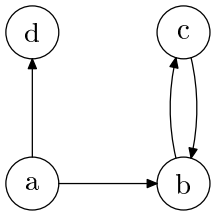
Harvard University
Computer Science 121
Midterm Solution 15-Oct-98

The points on the exam total 80. You may use any theorems proven in class or on the homeworks. Good luck!

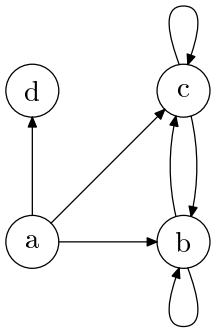
PROBLEM 1 (3+3+3 points)

Consider the following relation S:



(A) Is S reflexive? Symmetric? Transitive?
(B) Draw the transitive closure of S.
(C) Is the transitive closure of S reflexive? Symmetric? Transitive?

(A) S is neither reflexive nor symmetric nor transitive.
(B)

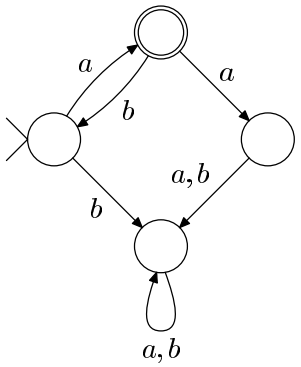


(C) The transitive closure of S (often denoted S^+) is, in this case, neither reflexive nor symmetric. It is, of course, transitive.

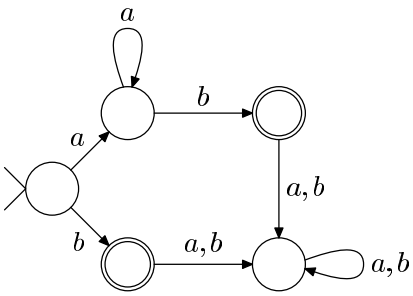
PROBLEM 2 (4+4+4+4 points)

For each of the DFAs drawn below, give both an informal description of and a regular expression for the language it accepts. Do not use the construction described in class: just build the regular expression from your informal description. Your descriptions should be short, and need not be backed by proofs.

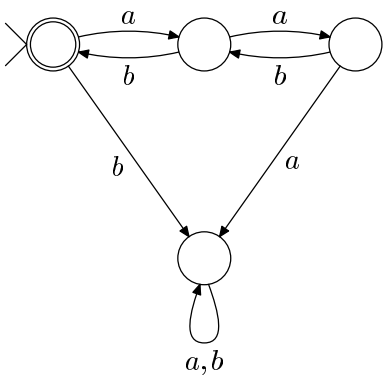
(A)



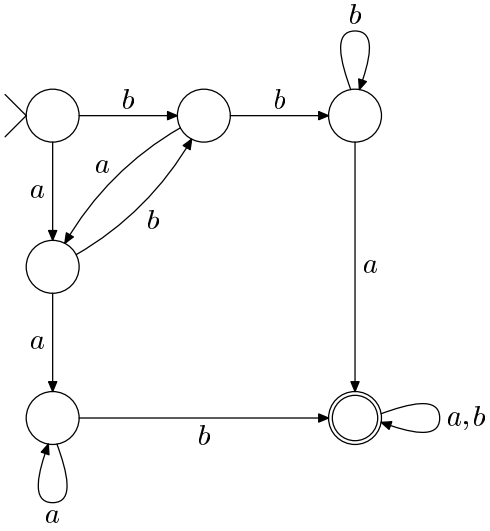
(B)



(C)



(D)



(A) The language accepted is the set of all strings starting with an a followed by any number of ba-pairs. Valid regular expressions include $a(ba)^*$ and $(ab)^*a$.

(B) The language accepted is any number of a's followed by a single b. The regular expression is a^*b. Note that this is not the smallest DFA that accepts this language.

(C) The language accepted is the set of strings w with the same number of a's and b's and the additional requirement that any prefix of w cannot have more b's than a's, nor can it have more than two more a's than b's. Some regular expressions are $(ab \cup aa(ba)^*bb)^*$ or $(ab \cup a(ab)^*b)^*$, or more concisely, just $(a(ab)^*b)^*$.

(D) The language accepted is all strings containing aab or bba as a substring. The regular expression is $(a \cup b)^*(aab \cup bba)(a \cup b)^*$.

PROBLEM 3 (12 points)

Let L be the language described by the regular expression (aab $\cup$ bba)*(ab)*.
Consider L', the language of all substrings of strings in L:
$L' = \{w : w \text{ is a substring of some } x \in L\}$. Write a regular expression corresponding to the language L'.

The easiest way to think about this is to think of the regular expression for L as the concatenation of the regular expressions for two other languages, L_1 and L_2, where the regular expression for L_1 is (aab $\cup$ bba)* and the regular expression for L_2 is (ab)*.

We will solve this problem by breaking it into three steps.

1. Clearly, any substring of L_1 is a substring of L, and is therefore in L'. So we need to find the regular expression for all substrings of $L_1 = \mathcal{L}((\text{aab} \cup \text{bba})^*)$.

   We note that any string in L_1 is a repetition of three-character long sequences. A substring of L_1 must start at either the beginning, middle, or end of one of these sequences, be followed by zero or more whole repetitions of such sequences, and terminate at either the beginning, middle, or end of such a sequence.

   Think about a long tape, on which is written one of these strings. We can produce any substring by cutting the tape at two points, letting the beginning and end fall off, retaining the middle as our substring. Our first cut must occur either at a boundary between three-letter sequences, meaning none of the previous sequence will show up in our substring, or in the middle of a three-letter sequence, meaning the end of that three-letter sequence will show up at the beginning our substring. In between our cuts will be zero or more whole three-letter sequences. Our second cut must occur at a boundary between three-letter sequences, meaning that none of the next seuqence will show up in our substring, or in the middle of a three-letter sequence, meaning that the beginning of that sequence shows up at the end of our substring. (Note that we produce the substring e by making both our cuts at the same point, or if our tape was empty to begin with.)

   Another way of saying this is that any substring must begin with a (possibly empty) suffix of aab or bba, be followed by any number of aabs or bbas in any order, and end with a (possibly empty) prefix of aab or bba.

   The regular expression for a suffix of aab or bba is (e $\cup$ b $\cup$ a $\cup$ ab $\cup$ ba).

   The regular expression for any number of aabs or bbas in any order is (aab $\cup$ bba)*.

   The regular expression for a prefix of aab or bba is (e $\cup$ a $\cup$ b $\cup$ aa $\cup$ bb).

   If we concatenate these three regular expressions together, we get our regular expression for all substrings of L_1: (e $\cup$ b $\cup$ a $\cup$ ab $\cup$ ba)(aab $\cup$ bba)*(e $\cup$ a $\cup$ b $\cup$ aa $\cup$ bb).

   (Note how the empty string falls out of the way we constructed the regular expression above.)

2. Also, any substring of L_2 is a substring of L, and is in L'. So we need to find the regular expression for all substrings of $L_1 = \mathcal{L}((\text{ab})^*)$. We will do this by following a logic similar to what we did in the last step.

   Any string in L_2 is just a sequence of abs repeated zero or more times. Any substring of L_2 must therefore begin with a suffix of ab, be followed by zero or more repetitions of ab, followed by a prefix of ab.

The regular expression for a suffix of ab is (e ∪ b).

The regular expression for zero or more repetitions of ab is (ab)*.

The regular expression for a prefix of ab is (e ∪ a).

If we concatenate these three regular expressions together, we get our regular expression for all substrings of L_2: (e ∪ b)(ab)*(e ∪ a).

3. We have now constructed regular expressions for substrings of L_1 and L_2, so now the only strings in L' that we haven't considered are those which "bridge the gap" between L_1 and L_2. That is, any string $w \in L$ can be written as $w = xy, x \in L_1, y \in L_2$. So far we have considered substrings of x and y, and we must now consider substrings which include some of x and some of y.

   We might think we can simply concatenate our regular expressions from the previous two steps. However, this does not work for two reasons. First of all, our first regular expression allows substrings of L_1 which do not end at a three-character sequence boundary, meaning they cannot possibly be followed by a substring of L_2. Secondly, our second regular expression allows substrings of L_2 which do not start at the beginning of a two-character boundary, meaning that they cannot possibly be preceded by a substring of L_1

   To explain this a different way, concatenating the two regular expressions would be saying that the concatenation of any substring of L_1 with a substring of L_2 is a substring of L_1L_2. For instance, concatenating our two regular expressions would tell us that that since aabbb is a substring of L_1 and b is a substring of L_2, then aabbbb must be a substring of L_1L_2. But this is not, in fact the case. The error here lies in picking a substring of L_1 which does not end at a valid ending for a string from L_1, and a second error lies in picking a substring of L_2 which does not begin with a valid beginning for a string from L_2.

   From this observation, it becomes clear that what we really want here is the concatenation of the regular expression for all substrings of L_1 which end at valid endings for strings in L_1 with the regular expression for all substrings which begin at valid beginnings for strings in L_2.

   The regular expression for all substrings of L_1 which end at valid endings for strings in L_1 is simply our answer from part 1, with the "prefix" part at the end removed:
   (e ∪ b ∪ a ∪ ab ∪ ba)(aab ∪ bba)*.

   The regular expression for all substrings which begin at valid beginnings for strings in L_2 is simply our answer from part 2, with the "suffix" part at the beginning removed:
   (ab)*(e ∪ a).

   So our regular expression for this "bridging the gap" strings is the concatenation of the above two regular expressions: (e ∪ b ∪ a ∪ ab ∪ ba)(aab ∪ bba)*(ab)*(e ∪ a).

Since all substrings of L must fall in one of the above three categories, our regular expression for L' is the union of the above regular expressions:

$$\begin{gathered}(\mathsf{e} \cup \mathsf{b} \cup \mathsf{a} \cup \mathsf{ab} \cup \mathsf{ba})(\mathsf{aab} \cup \mathsf{bba})^*(\mathsf{e} \cup \mathsf{a} \cup \mathsf{b} \cup \mathsf{aa} \cup \mathsf{bb}) \\ \cup (\mathsf{e} \cup \mathsf{b})(\mathsf{ab})^*(\mathsf{e} \cup \mathsf{a}) \\ \cup (\mathsf{e} \cup \mathsf{b} \cup \mathsf{a} \cup \mathsf{ab} \cup \mathsf{ba})(\mathsf{aab} \cup \mathsf{bba})^*(\mathsf{ab})^*(\mathsf{e} \cup \mathsf{a})\end{gathered}$$

Many of you thought that L' should be the language of all strings with not more than three a's or b's in a row. However, this is not the case: bbabaa is an example of such a string which is not in L'.

PROBLEM 4 (4+4+4+4 points)

True or False? Justify your answers.

(A) If $L_1^* = L_2^*$ then it must be that $L_1 = L_2$.

(B) For every finite language L, $L^* \neq L$.

(C) The number of regular languages over $\{a, b\}$ is countably infinite.

(D) An intersection of countably many finite regular languages is necessarily regular.

(A) **False.** A good counterexample is $L_1 = \{a\}$ and $L_2 = \{a, aa\}$. In this case, $L_1^* = L_2^* = \{a^n : n \geq 0\}$, but $L_1 \neq L_2$. More generally, we could also say that if L is any language for which $L \neq L^*$, the claim fails for $L_1 = L$ and $L_2 = L^*$ (because $(L^*)^* = L^*$ for any language).
(B) **False.** Let $L = \{e\}$. Then $L^* = \{e\}$ also, so that $L = L^*$. L is certainly finite; it has only one element.
(C) **True.** Each regular language can be described by a regular expression, which is a string over the alphabet $\{a, b, \emptyset, (,), *\}$. Since there are only countably many strings over any alphabet, there are only countably many regular expressions for languages in $\{a, b\}^*$, so there can be at most countably many regular languages. On the other hand, since there are a countably infinite number of strings over $\{a, b\}^*$, and for every string $x \in \Sigma^*$ the language $\{x\}$ is regular (because it is finite) and distinct, there are an infinite number of regular languages. These two facts together establish the claim.
(D) **True.** Let $L_1, L_2, \ldots$ be the languages being intersected. The result of the intersection (let us call it L_∞) can contain no strings not in L_1, so that $L_\infty \subseteq L_1$. Since L_1 is finite, so therefore is any subset of it, including L_∞. Since L_∞ is finite, it is regular.

PROBLEM 5 (12 points)

Let $f : X \to Y$ be a function from a set X to a set Y, and let $R \subseteq Y \times Y$ be a relation on Y. Define the relation $S \subseteq X \times X$ on the set X by: $(x, y) \in S \iff (f(x), f(y)) \in R$. Prove that if R is symmetric, then S is symmetric.

Despite all of the notation and given information, the problem statement is relatively simple: Show that if R is symmetric, then so is S. (Note that this has the general form: If A, then B.) We begin by assuming that R is symmetric (i.e., we take it as a *given fact* that R is symmetric). We must then show that S is symmetric. To prove that S is symmetric, we go back to the definition: S is symmetric if *for any* $(x, y) \in S$, then $(y, x) \in S$. We prove this by assuming that (x, y) is some *arbitrary* element of S and then showing that (y, x) must also be in S. Beginning with $(x, y) \in S$, we use the given information to translate this into a statement about R: $(f(x), f(y)) \in R$. We then use the symmetry of R to get that $(f(y), f(x)) \in R$, and again use the given information to translate this back into a statement about S: $(y, x) \in S$, which is the desired conclusion.

More concisely: Suppose R is symmetric. We now show that S is symmetric:

Suppose (x, y) is some arbitrary element of S.

Then $(f(x), f(y)) \in R$, by the definition of S.

Then $(f(y), f(x)) \in R$, since R is symmetric.

Then $(y, x) \in S$, by the definition of S. $\square$

Comments. Although there is an *if and only if* relationship between $(x, y) \in S$ and $(f(x), f(y)) \in R$, there is *not* necessarily a bijection between $(x, y) \in S$ and $(a, b) \in R$. For example, if the function f is not *one-to-one*, there may be several $(x_i, y_i) \in S$ that correspond to the same $(a, b) \in R$; and if f is not *onto*, there may not be *any* $(x, y) \in S$ that corresponds to a given $(a, b) \in R$. (Consider the case where $f(x) = 0$ for all $x \in X$, which is neither one-to-one nor onto.) So it is very important that in the above proof we begin with an arbitrary pair $(x, y) \in S$, rather than starting with some arbitrary pair $(a, b) \in R$.

PROBLEM 6 (6+9 points)

(A) Write the 5-tuple for the DFA given in Problem 2, Part D.

(B) Draw a DFA accepting the language $L = \{w \in \{a,b\}^* : ababb \text{ is a substring of } w\}$.

(A)

$$\begin{aligned} M &= (K, \Sigma, \delta, s, F) \\ K &= \{q_1, q_2, q_3, q_4, q_5, q_6\} \\ \Sigma &= \{a, b\} \\ s &= q_1 \\ F &= \{q_6\} \end{aligned}$$

and δ is given by

p	σ	$\delta(p,\sigma)$	p	σ	$\delta(p,\sigma)$
q_1	a	q_4	q_1	b	q_2
q_2	a	q_4	q_2	b	q_3
q_3	a	q_6	q_3	b	q_3
q_4	a	q_5	q_4	b	q_2
q_5	a	q_5	q_5	b	q_6
q_6	a	q_6	q_6	b	q_6

(B)