CS 373, Spring 2009. Solutions to Mock midterm 1 (Based on first midterm in CS 273, Fall 2008.)

Problem 1: Short Answer (8 points)

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

(a) If an NFA M accepts the empty string (i.e., ϵ), does M's start state have to be an accepting state? Why or why not?

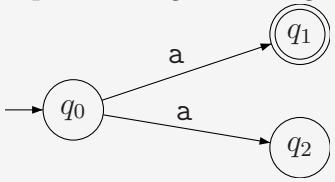
Solution: No it does not have to be an accepting state. Because M is an NFA, we can accept the empty string by having a non-accepting start state which has an ϵ-transition (or a sequence of such ϵ-transitions) to an accepting state.

(b) Is every finite language regular? Why or why not?

Solution: Yes. Every finite language is regular. You can build an NFA for it by building a linear DFA that recognizes each string individually. Then join them all to a common start state using ϵ-transitions.

(c) Suppose that an NFA $M = (Q, \Sigma, \delta, q_0, F)$ accepts a language L. Create a new NFA M' by flipping the accept/non-accept markings on M. That is, $M' = (Q, \Sigma, \delta, q_0, Q - F)$. Does M' accept $\overline{L}$ (the set complement of L)? Why or why not?

Solution: This only works for a DFA. Consider the following NFA. It accepts the language $\{\mathtt{a}\}$. If you flip the accept markings, it recognizes $\{\mathtt{a}, \epsilon\}$.



(d) Simplify the following regular expression $\emptyset^*(\mathtt{a} \cup \mathtt{b}) \cup \emptyset\mathtt{b}^* \cup \epsilon\mathtt{abb}$.

Solution: This is equal to $\epsilon(\mathtt{a} \cup \mathtt{b}) \cup \emptyset \cup \mathtt{abb}$ which is just $\mathtt{a} \cup \mathtt{b} \cup \mathtt{abb}$. This can also be written as $\mathtt{a} + \mathtt{b} + \mathtt{abb}$.

Problem 2: DFA design (6 points)

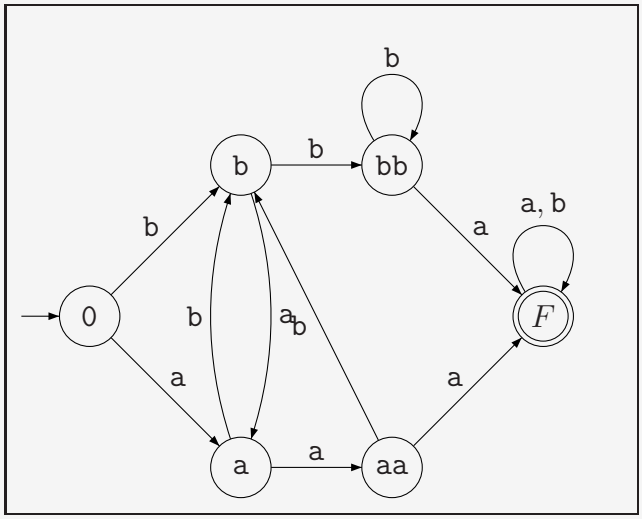
Let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$. Let L be the set of strings in Σ^* which contain the substring `bba` or the substring `aaa`.

For example, `aabba` $\in L$ and `baaab` $\in L$, but `babab` $\notin L$. Strings shorter than three characters are never in L.

Construct a DFA that accepts L and give a state diagram showing **all** states in the DFA.

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

Solution:



Problem 3: Remembering definitions (8 points)

(a) Define formally what it means for a DFA $(Q, \Sigma, \delta, q_0, F)$ to accept a string $w = w_1 w_2 \dots w_n$.

Solution: The DFA accepts w if there is a state sequence $s_0 s_1 \dots s_n$ such that

- $s_0 = q_0$
- $s_n \in F$, and
- $s_i = \delta(s_{i-1}, w_i)$ for every $i \in [i, n]$.

(b) Let Σ and Γ be alphabets. Suppose that h is a function from Σ^* to Γ^*. Define what it means for h to be a ***homomorphism***.

Solution: A mapping h is a homomorphism if $h(xy) = h(x)h(y)$ for any strings x and y.

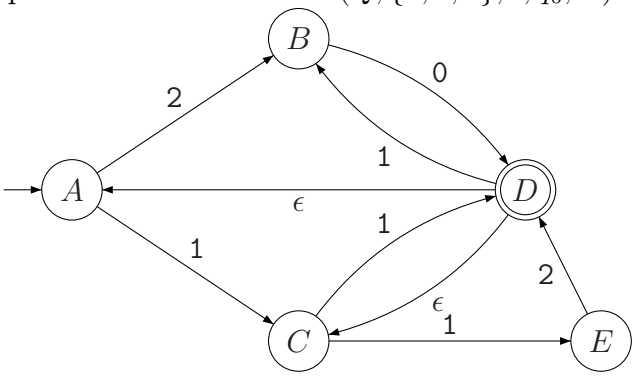
Or, equivalently, h is a homomorphism if $h(c_1 c_2 \dots c_n) = h(c_1)h(c_2)\dots h(c_n)$ for any sequence of characters $c_1 c_2 \dots c_n$.

Or, you could say it in words: the output of h on a string is the concatenation of its outputs on the individual characters making up the string.

Or you could even say: h is a homomorphism if it operates on strings character-by-character.

Problem 4: NFA transitions (6 points)

Suppose that the NFA $N = (Q, \{0, 1, 2\}, \delta, q_0, F)$ is defined by the following state diagram:



Fill in the following values:

(a) $F =$

Solution: $F = \{D\}$.

(b) $\delta(A, 0) =$

Solution: $\delta(A, 0) = \emptyset$

(c) $\delta(C, 1) =$

Solution: $\delta(C, 1) = \{D, E\}$

(d) $\delta(D, 1) =$

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

(e) List the members of the set $\{q \in Q \mid D \in \delta(q, 2)\}$

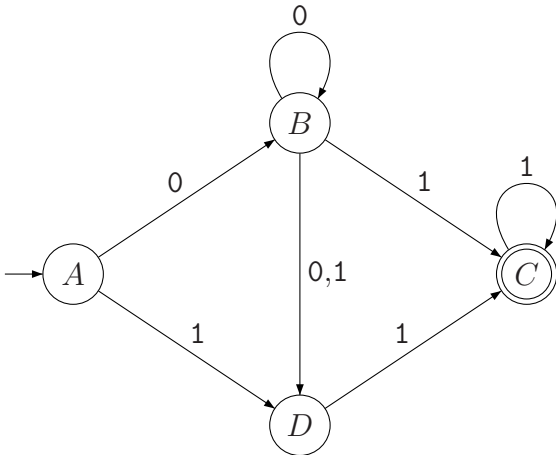
Solution: E.

(f) Does the NFA accept the word 11120? (Yes / No)

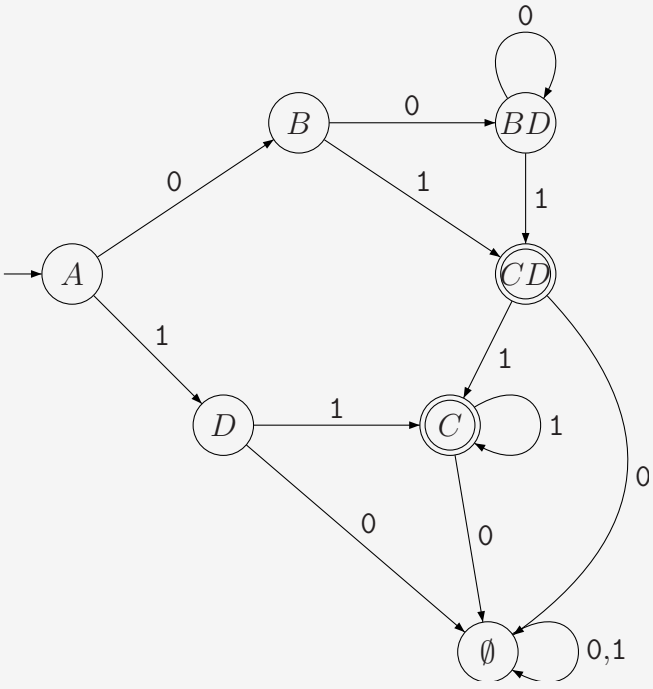
Solution: Yes. The state sequence is $ACDCDABD$.

Problem 5: NFA to DFA conversion (6 points)

Convert the following NFA to a DFA recognizing the same language, using the subset construction. Give a state diagram showing all states reachable from the start state, with an informative name on each state. Assume the alphabet is $\{0, 1\}$.



Solution:



Problem 6: Short Construction (8 points)

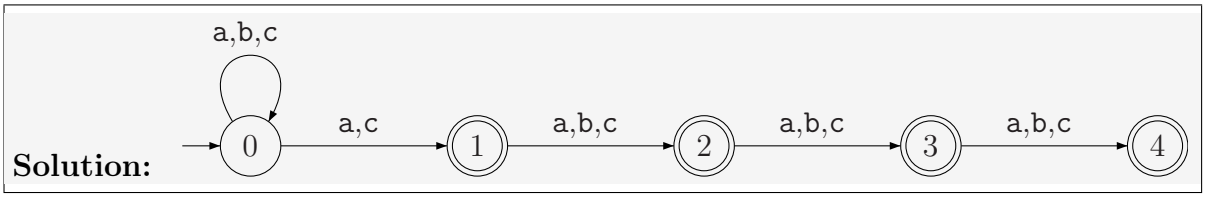
(a) Give a regular expression for the language L containing all strings in $\mathtt{a}^*\mathtt{b}^*$ whose length is a multiple of three. E.g. L contains `aaaabb` but does not contain `ababab` or `aaabb`.

Solution: $(\mathtt{aaa})^*(\mathtt{bbb})^* \cup (\mathtt{aaa})^*\mathtt{aab}(\mathtt{bbb})^* \cup (\mathtt{aaa})^*\mathtt{abb}(\mathtt{bbb})^*$.

(b) Let $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$. Give an NFA for the language L containing all strings in Σ^* which have an `a` or a `c` in the last four positions. E.g. `bbabbb` and `abbbcb` are both in L, but `acabbbb` is not. Notice that strings of length four or less are in L exactly when they contain an `a` or a `c`.

You will receive **zero** *credit if your NFA contains more than 8 states.*

Solution:



Problem 7: NFA modification and tuple notation (8 points)

For this problem, the alphabet is always $\Sigma = \{a, b\}$.

Given an NFA M that accepts the language L, design a new NFA M' that accepts the language $L' = \{twt \mid w \in L, t \in \Sigma\}$. For example, if aab is in L, then $aaaba$ and $baabb$ are in L'.

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

Solution: The new NFA M' has two copies of M's states, plus new initial and final states. Copy A is for the strings starting in `a` and copy B is for the strings starting in `b`. When we read the first character, we transition to the start state of the appropriate copy. When we reach a final state in our copy, there is a transition from that final state to the new final state, consuming the appropriate input character. E.g. the final states of copy A have a transition on a into the new final state.

(b) Suppose that $M = (Q, \Sigma, \delta, q_0, F)$. Give the details of your construction of M', using tuple notation.

Solution:

$M' = (Q', \Sigma, \delta', q_S, \{q_F\})$ where $Q' = \{q_S, q_F\} \cup \{q^A \mid q \in Q\} \cup \{q^B \mid q \in Q\}$ and δ' is defined as follows:

$$\delta'(q_S, a) = q_0^A$$
$$\delta'(q_S, b) = q_0^B$$
$$\delta'(q^A, t) = \{r^A \mid r \in \delta(q, t)\} \text{ if } q \in Q - F, \text{ or if } q \in F \text{ and } t = b$$
$$\delta'(q^B, t) = \{r^B \mid r \in \delta(q, t)\} \text{ if } q \in Q - F, \text{ or } q \in F \text{ and } t = a$$
$$\delta'(q^A, a) = \{q_F\} \cup \{r^A \mid r \in \delta(q, t)\} \text{ for every } q \in F$$

$\delta'(q^B, b) = \{q_F\} \cup \{r^B \mid r \in \delta(q, t)\}$ for every $q \in F$

$\delta'(q, t) = \emptyset$ for all other inputs

Notice that the old final states in each copy of M need to keep all their old transitions and also add the transition into the new final state.

Full credit didn't require getting absolutely every detail correct.

The big mistake many people made was to use a single copy of the original set of states, to which they just added new start and end states to M. This doesn't allow the NFA to "remember" the first character of the string, so it can't verify that the last character is the same. Reasonably well-formed versions of this answer were worth 2/8 points.