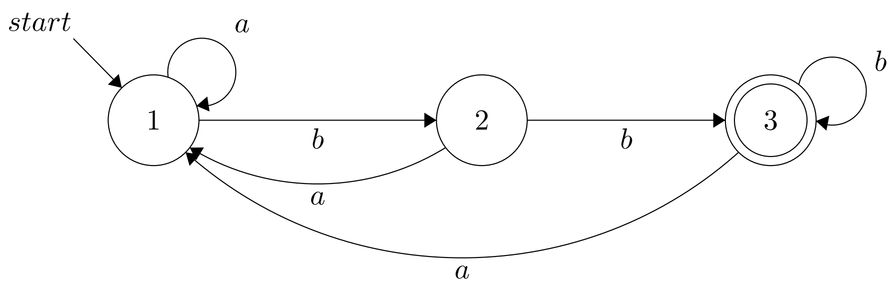
Name and St.ID#:____

CSE396 Prelim I Answer Key Spring 2017

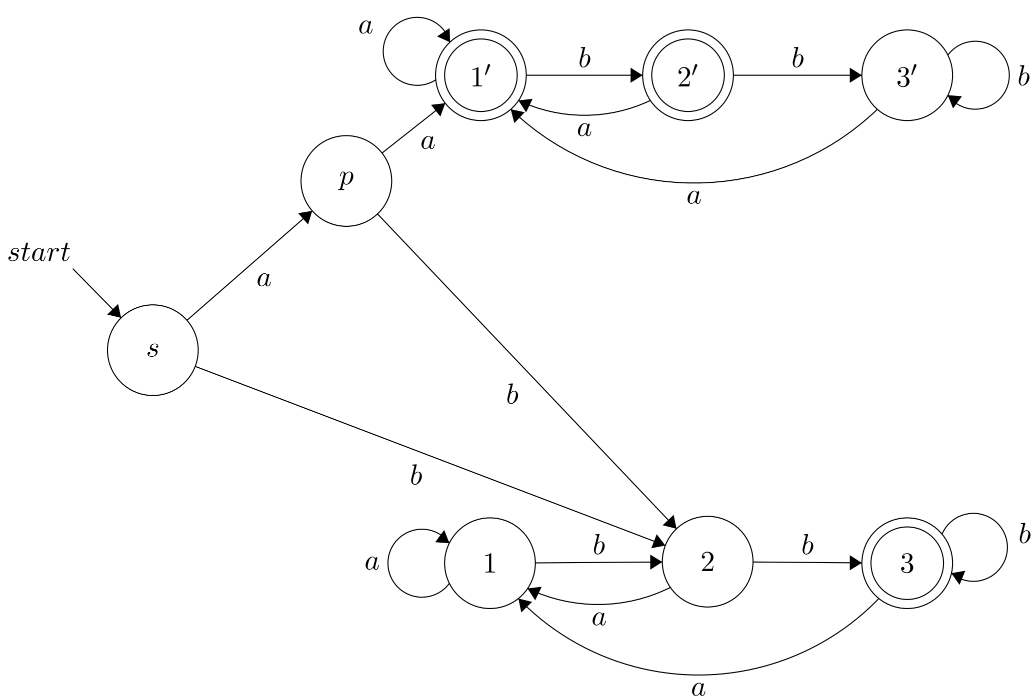
(1) (24 pts.)

Define A to be the language of strings $x \in \{a,b\}^*$ such that x either begins with aa or ends with bb, but not both. Design a DFA M such that $L(M) = A$. A node-arc diagram that shows the start and final states clearly is good enough, plus the correctness of your M must be clear either from your theoretical technique or from strategic comments on how M works.

Answer: First let us design a DFA M_1 for the "ends with bb" property:



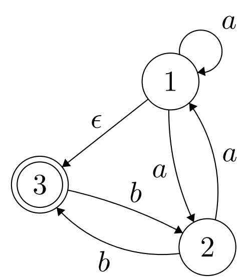
Now take M_1' to be complementary machine in which states 1 and 2 are accepting and state 3 is not. We design M to have two non-accepting states s and p plus a copy of each machine, for 8 states in all. At s, if the first char of the input x is a b then we know it didn't start with aa, so by the XOR we need x to end in bb. Since we already have one b, the arc goes to state *2* of M_1, not state 1. If x begins with an a then we go to state p. At p, if we then get a b we know again that x doesn't start with aa, and we're actually in the same situation as when x begins with b, so we go to state 2 of M_1. If we get an a at state p, then that's the second a at the start, so we shift mode into *not* wanting to end in bb. So we go to the start state $1'$ of the *complementary* machine M_1'. That completes M such that $L(M) = A$ and also its verification:



The other good way was to draw a second small DFA M_2 such that $L(M_2) = aa(a \cup b)^*$ and then make M to be the Cartesian product of M_1 and M_2 using XOR. M_2 is in some ways simpler to draw than M_1 but it has a dead state which technically makes 4 states total—and since the processing has to go on even after the "aa" part is dead you can't make it a dead state for all of M. So you're doing a 3×4 Cartesian product which could make you liable for 12 states, but if you carry it out in an economical "as-needed" manner then you get the same 8 states as above. Let's call the states of M_2 as s_2, q_2, r_2, d_2 to go with $1, 2, 3$ for M_1 as above. With Cartesian labels, the states $1, 2, 3$ in the above diagram become $(1, d_2), (2, d_2), (3, d_2)$ since they are paired with the dead state of M_1. And $s = (1, s_2)$ since it is the start state of both machines, and $p = (1, q_2)$. Then $1' = (1, r_2)$ and since r_2 is a "nirvana" state of M_2, the remaining labels are $2' = (2, r_2)$ and $3' = (3, r_2)$. The final states are $(3, d)$, $(1, r_2)$, and $(2, r_2)$, being the reachable states that have either 3 from M_1 or r_2 from M_2, but not both.

(2) (18 + 3 + 6 = 27 pts.)

Let $N = (Q, \Sigma, \delta, s, F)$ be the NFA with $Q = \{1, 2, 3\}$, $\Sigma = \{a, b\}$, $s = 1$, $F = \{3\}$, and δ given by the arcs $(1, \epsilon, 3)$, $(1, a, 1)$, $(1, a, 2)$, $(2, a, 1)$, $(2, b, 3)$, and $(3, b, 2)$, shown by:



(a) Calculate a DFA M such that $L(M) = L(N)$ (no "comments" needed if the method is clear).

(b) Find a string x such that N can process x from 1 to any one of its three states—figuratively speaking, such that x "lights up" all three states of N.

(c) Are there strings w that N cannot process at all, so that N "dies"? Most in particular, can w have the form xz—that is, begin with your string x from part (b) which turned all states on?

Answer: The ϵ-arc gives us "Whenever 1, then also 3" and in particular makes the start state of the DFA be $\{1, 3\}$ not just $\{1\}$. This is crucial because ϵ belongs to $L(N)$ but we need the "3" to recognize that the start state of M is accepting. Starting this way also lets us henceforth only have to worry about *trailing* ϵ's by the "Roman soldier" reasoning in lecture. We can thus make a table that is IMHO more useful than what the text does:

$$\begin{array}{ll} \underline{\delta}(1, a) = \{1, 2, 3\} & \underline{\delta}(1, b) = \emptyset \\ \underline{\delta}(2, a) = \{1, 3\} & \underline{\delta}(2, b) = \{3\} \\ \underline{\delta}(3, a) = \emptyset & \underline{\delta}(3, b) = \{2\}. \end{array}$$

Doing a breadth-first search from $S = \{1, 3\}$ then gives us:

$$\begin{aligned} \Delta(S, a) &= \underline{\delta}(1, a) \cup \underline{\delta}(3, a) = \{1, 2, 3\} \cup \emptyset = \{1, 2, 3\} \\ \Delta(S, b) &= \underline{\delta}(1, b) \cup \underline{\delta}(3, b) = \emptyset \cup \{2\} = \{2\}. \end{aligned}$$

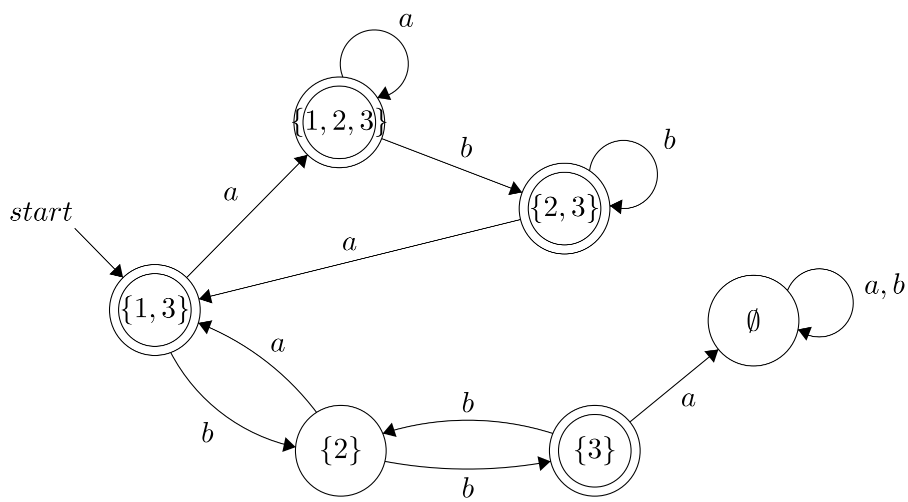
Notice we already "lit all the lights" on an a. Since we got there on an a, we automatically know that $\Delta(\{1,2,3\},a) = \{1,2,3\}$, but this need not be true of $\Delta(\{1,2,3\},b)$. Indeed, we get

$$\Delta(\{1,2,3\},b) = \underline{\delta}(1,b) \cup \underline{\delta}(2,b) \cup \underline{\delta}(1,b) = \emptyset \cup \{3\} \cup \{2\} = \{2,3\}$$

only, which is what begins the long decline and ultimate fall of the Nondeterministic Empire. Soldiering on with breadth-first search, we need to expand the new states $\{2\}$ and $\{2,3\}$. We get:

$$\begin{aligned}
\Delta(\{2\},a) &= \underline{\delta}(2,a) = \{1,3\} \qquad \text{(back to start)}\\
\Delta(\{2\},b) &= \underline{\delta}(2,b) = \{3\} \qquad \text{(but this counts as new)}\\
\Delta(\{2,3\},a) &= \underline{\delta}(2,a) \cup \underline{\delta}(3,a) = \{1,3\} \cup \emptyset = \{1,3\}\\
\Delta(\{2,3\},b) &= \underline{\delta}(2,b) \cup \underline{\delta}(3,b) = \{3\} \cup \{2\} = \{2,3\} \qquad \text{(not new)}\\
\Delta(\{3\},a) &= \underline{\delta}(3,a) = \emptyset \qquad \text{(our 6th state but it's dead)}\\
\Delta(\{3\},b) &= \underline{\delta}(3,b) = \{2\} \qquad \text{(not new).}
\end{aligned}$$

Since we automatically know $\Delta(\emptyset,a) = \Delta(\emptyset,b) = \emptyset$, we're done—we've closed the machine M. The final states of M are "everything with 3" so we get:



Note we already answered part (b) with $x = a$. For (c) note from the DFA diagram that there is a long and tortuous path from the "all-on" state $\{1,2,3\}$ to the dead state in five steps processing $z = babba$. This makes $xz = ababba$ unprocessable by N. The shortest "dying string" from the start is bba, but $ababba$ has the extra "insult-plus-injury" of first turning all the states on. (Grading was 4 points for $w = bba$ but $xz = ababba$ (or the equivalent) needed for the full 6 points.)

(3) ($5 \times 5 = 25$ pts.) *Multiple Choice.*

Note the first three questions refer to the NFA N in problem (2).

1. When we eliminate state 2 from the NFA N in problem (2), we need to update which items involving only states 1 and 3:
   (a) Just the two arcs in the diagram, $(1,a,1)$ and $(1,\epsilon,3)$.
   (b) Besides those two arcs, we need to add a self-loop $(3,bb,3)$, but nothing from 3 back to 1.
   (c) Because state 2 has incoming and outgoing arcs from both 1 and 3, we need to update $2 \times 2 = 4$ entries.
   (d) Because of the ϵ-arc, we can just combine the states 1 and 3 together, getting a simple 1-state GNFA after eliminating state 2.

Answer: (c).

2. Regarding the languages $L_{1,1}$ and $L_{1,3}$ of N—whether before or after eliminating state 2 it doesn't matter—which of the following is true for this particular machine?

   (a) $L_{1,1} \subseteq L_{1,3}$.

   (b) $L(N) = L_{1,1} \cup L_{1,3}$.

   (c) $L_{1,1}$ contains $(a \cup aa)^*$, which simplifies to a^*.

   (d) All of the above.

   Answer: (d) All of the above. Part (a) comes because of the ϵ-arrow—it isn't true in general—and (b) follows from that and the fact that $L(N) = L_{1,3}$ by definition. When you carry out the step of of eliminating state 2 above, you get a loop on aa as well as the original loop on a at state 1. Since the aa-loop can be simulated by two go-rounds of the a-loop, which is another way of saying $(aa)^* \subset a^*$, you can just discard it, which means $(a \cup aa)^* = a^*$. And a^* was already part of $L_{1,1}$.

3. A valid regular expression for $L(N)$ is:

   (a) $(aa \cup a \cup (ab \cup \epsilon)(bb)^*ba)^*(\epsilon \cup ab)(bb)^*$.

   (b) $(a \cup b)^*$ since state 1 could be accepting too.

   (c) $(aa \cup ab \cup ba \cup bb)^*$ since N accepts all even-length strings.

   (d) $(a \cup ab(bb)^*ba)^*ab(bb)^*$.

   Answer: Without going through the whole NFA-to-regesx exercise, we can eliminate (b) because $(a \cup b)^*$ is all strings and we know there are strings that N rejects. And we can eliminate (c) because we found an odd-length string that N accepts (and N doesn't accept all even-length strings anyway, it's "fake news"). That leaves (a) versus (d), both of which look "close." Well, the trouble with (d) is that the 'ab' part in the middle is made *mandatory*, which rules out not only ϵ but also the all-lights-on string a as possibilities. Hence it's too strict. Item (a) happens to be right—but there are a bunch of other right ones including taking the hint from 2(c) and simplifying it to $(a \cup ab(bb)^*ba)^*(\epsilon \cup ab)(bb)^*$.

4. The symmetric difference of a language A with its complement $\tilde{A}$ in Σ^* always equals:

   (a) $\emptyset$.

   (b) $\{\epsilon\}$.

   (c) A.

   (d) A^*—no wait!—Σ^*.

   Answer: Well, I intended to make a question whose answer was $\emptyset$, but this wasn't it—correct is Σ^*. One might say (d) was "closest" but CSE396 isn't horseshoes or atom bombs, so any answer had to be accepted and this became a free 5 points.

5. For a general language L, the relation $x \approx_L y$ when $xy \in L$ is:

   (a) An equivalence relation.

   (b) Reflexive and symmetric, but not transitive.

   (c) Symmetric, but not reflexive or transitive.

   (d) Neither reflexive, symmetric, nor transitive (in general).

Answer: Consider $L = \{ab, baa\}$. Since L has no double words, there is *no* string x such that $xx \in L$, so $x \approx_L x$ never happens, so the relation is as irreflexive as can be. Nor is it symmetric, because $a \approx_L b$ but not vice-versa (likewise $ba \approx_L a$ but not vice-versa). This already torpedos (a,b,c) but let's consider "transitive": We have $a \approx_L b$ and $b \approx_L aa$, but not $a \approx_L aa$ because $aaa \notin L$. So the answer is (d). Very few got this; so many said (a) that this became an instance of bewaring "rosy expectations" in favor of colf logic—which is one of the ulterior messages of the course. [This, the "xz" part of 2(c), part 3(c) above, and getting problem 4 exactly right were the main "A-level points" on the grading scheme, plus the 3 points for comments/strategy in problem 1. In fact, problem 1 had scores averaging 2-3 points lower than expected, but the freebie on 3(d) offset this so the difficulty tuning for the pre-set curve was pretty-much on-target.]

(4) (24 pts.)

Over $\Sigma = \{0, 1\}$, define $L = \{\, x : \#00(x) = \#10(x) \,\}$. Recall from Problem Set 4 that for any $u, x \in \Sigma^*$, $\#u(x)$ is the number of times u occurs as a substring of x, counting occurrences that overlap. Prove via the Myhill-Nerode technique that L is not a regular language.

Answer: Take $S = (00)^*$, which is clearly infinite. Let any $x, y \in S$ $(x \neq y)$ be given. Then there are numbers $m, n \geq 0$ such that $x = (00)^m$ and $y = (00)^n$. (We could also postulate $m < n$ without loss of generality but we won't need this.) Consider taking, ummm..., $z = (10)^m$ Then $xz = (00)^m(10)^m$ which looks like it is in L, but wait—because of the overlaps a string like $w = (00)^2(10)^2 = 00001010$ has $\#00(w) = \#00(x) = 3$, not 2, which $\neq \#10(w)$. So we actually need to take $z = (10)^{2m-1}$ to make $xz \in L$. Then $yz = (00)^n(10)^{2m-1}$ and since $\#00(yz) = 2n - 1 \neq 2m - 1 = \#10(yz)$, we do get $yz \notin L$. Since $x, y \in S$ are arbitrary, S is PD for L, and since S is infinite, L is not regular by the Myhill-Nerode Theorem.

Except there's one little niggle: taking $S = (00)^*$ allows $m = 0$, but what does "$z = (10)^{2m-1}$" mean when $m = 0$? It says $(10)^{-1}$ but we don't have negative powers of strings. If we stipulate that it means ϵ then we're still OK since then $xz = \epsilon \cdot \epsilon = \epsilon$ and $\epsilon \in L$. But we can totally avoid this "obnoxious edge case" by taking $S = (00)^+$ instead. Then we get $m, n \geq 1$ and the above proof becomes airtight.

We can also take $S = 0^+ = 00^*$ (note that 00^* is **not** the same as $(00)^*$, which was an unforseen error). Then when we let any $x, y \in S$ be given we get $x = 0^{m+1}$ and $y = 0^{n+1}$ with $n \neq m$. Then we can exactly say $\#00(x) = m$ and take $z = (01)^m$ after all. This was another completely-correct answer given by several. [Most, however, did the simple $xz = (00)^m(10)^m$ thing and fell victim to optical appearances to say $xz \in L$, forgetting the overlaps. This was a 4-point deduction, mitigated to -3 if there was *some* attempt to justify with "because" and something tangible, not just saying "$xz \in L$" and moving on. The A-level points were reckoned as 3 on problem 1, 2 on 2, 8 for 3(c) and 3(e)—not 10 because 2 pts. part credit was given for answering "d" on 3(c)—and 4 here, making 17, but I figured problem 1 brought 3 more because it was a little longer than comparable problems in past years, so that made the usual 20 out of 100.]

END OF EXAM