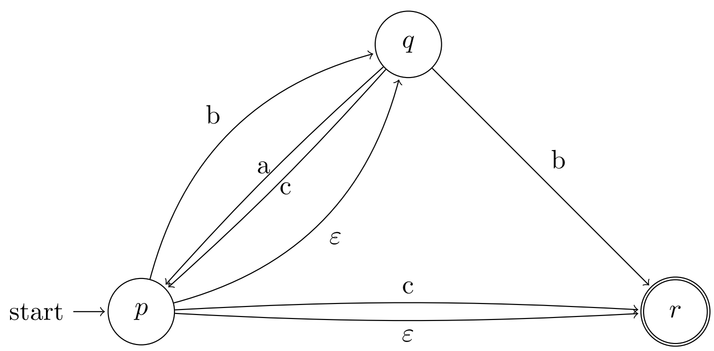
Nathan Bickel, Jackson Ginn, Jack Hyatt, Chance Storey
CSCE 355: Section 001
Professor: Dr. Fenner
February 13, 2024

CSCE 355 Homework 3

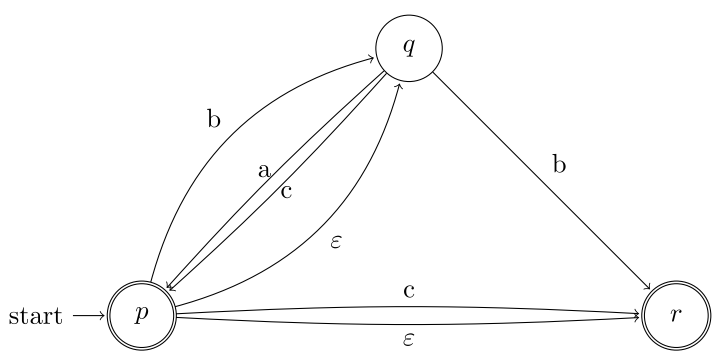
Problem 1 For the ε-NFA of textbook Exercise 2.5.2,

	ε	a	b	c
$\rightarrow p$	$\{q, r\}$	$\emptyset$	$\{q\}$	$\{r\}$
q	$\emptyset$	$\{p\}$	$\{r\}$	$\{p, q\}$
$*r$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

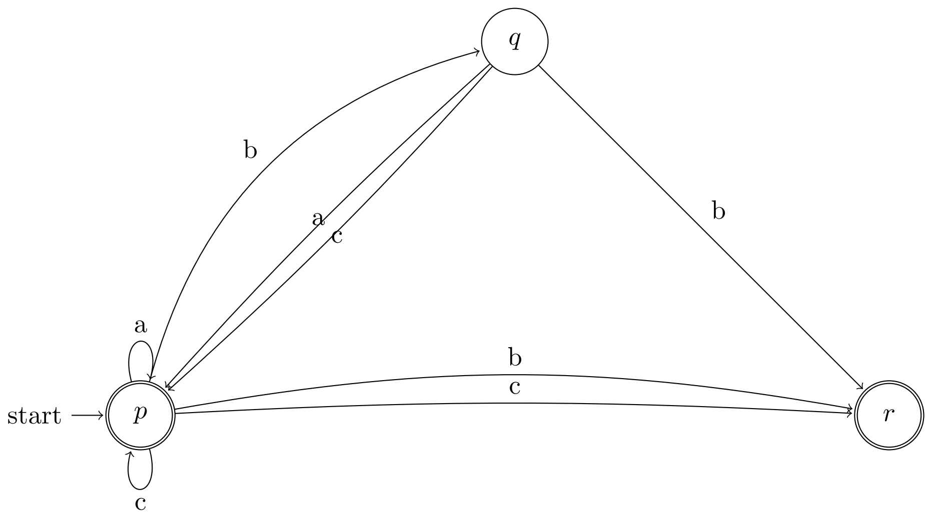
find an equivalent NFA (without ε-moves) using the method explained in class. This is also Method 2 described in the COURSE NOTES (link from the course homepage) in Section 10.4.



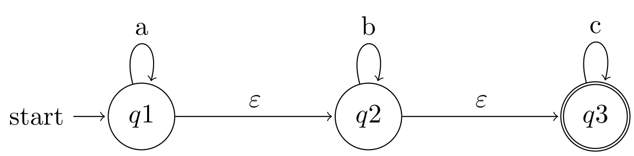
After making p accepting as it has an ε-transition to r, an accepting state:



After adding new transitions to bypass ε-transitions:



Problem 2 Do Problem 2.3 (pp. 81–82): Design an ε-NFA for the following language: the set of all strings consisting of zero or more a's followed by zero or more b's, followed by zero or more c's. Try to use ε-transitions to simplify your design.



Problem 3 Do Problem 2.3 (pp. 81–82). Here is the transition function of a simple, deterministic automaton with start state A and accepting state B:

	0	**1**
A	A	B
B	B	A

We want to show that this automaton accepts exactly those strings with an odd number of 1's, or more formally:

$$\delta(A, w) = B \text{ if and only if } w \text{ has an odd number of 1's.}$$

Here, δ is the extended transition function of the automaton; this is, $\delta(A, w)$ is the state that the automaton is in after processing input string w. The proof of the statement above is an induction on the length of w. Below, we give the proof with reasons missing. You must give a reason for each step, and then demonstrate your understanding of the proof by classifying your reasons into the following three categories:

A) Use of the inductive hypothesis.

B) Reasoning about properties of deterministic finite automata, e.g., that if string $s = yz$, then $\delta(q, s) = \delta(\delta(q, y), z)$.

C) Reasoning about properties of binary strings (strings of 0's and 1's), e.g. that every string is longer than any of its proper substrings.

Basis ($|w| = 0$):

1. $w = \varepsilon$ because:
2. $\delta(A, \varepsilon) = A$ because:
3. ε has an even number of 0's because:

Induction ($|w| = n > 0$):

4. There are two cases: (a) when $w = x1$ and (b) when $w = x0$ because:

Case (a):

5. In case (a), w has an odd number of 1's if and only if x has even number of 1's because:
6. In case (a), $\delta(A, x) = A$ if and only if w has an odd number of 1's because:
7. In case (a), $\delta(A, w) = B$ if and only if w has an odd number of 1's because:

Case (b):

8. In case (b), w has an odd number of 1's if and only if x has an odd number of 1's because:
9. In case (b), $\delta(A, x) = B$ if and only if w has an odd number of 1's because:
10. In case (b), $\delta(A, w) = B$ if and only if w has an odd number of 1's because:

---

1. (C): ε is the unique string with length 0.
2. (B): If no characters are processed, then clearly the state will stay the same.
3. (C): ε has 0 characters, so in particular it has zero 0's. Since zero is even, ε has an even number of 0's.
4. (C): By definition, since $|w| > 0$, w has a primary prefix x and a last character $a \in \Sigma$. Since $\Sigma = \{0, 1\}$, there are two options for the last character. These are the two cases.
5. (C): Since $w = x1$ has one more 1 than in w, the number of 1's in w and x must have different parities (adding 1 to any number flips the parity).
6. (A): We have

$$\begin{aligned}
\delta(A, x) = A &\iff \delta(A, x) \neq B && \text{(there are only two states)} \\
&\iff x \text{ does not have an odd number of 1's} && \text{(induction hypothesis)} \\
&\iff x \text{ has an even number of 0's} && \text{(integer property)} \\
&\iff w \text{ has an odd number of 1's.} && \text{(from 5)}
\end{aligned}$$

7. (B): We have

$$\begin{aligned}
\delta(A, w) = B &\iff \delta(\delta(A, x), 1) = B && \text{(since } w = x1\text{)} \\
&\iff \delta(A, x) = A && \text{(definition of } \delta\text{: } \delta(q, 1) = B \iff q = A\text{)} \\
&\iff w \text{ has an odd number of 1's.} && \text{(from 6)}
\end{aligned}$$

8. (C): Since $w = x0$ has the same number of 1's as w, the number of 1's in w and x must have the same parity as they are equal.

9. (A): We have

$$\begin{aligned} \delta(A,x) = B &\iff x \text{ has an odd number of 1's} && \text{(induction hypothesis)} \\ &\iff w \text{ has an odd number of 1's.} && \text{(from 8)} \end{aligned}$$

10. (B): We have

$$\begin{aligned} \delta(A,w) = B &\iff \delta(\delta(A,x),0) = B && \text{(since } w = x0\text{)} \\ &\iff \delta(A,x) = B && \text{(definition of } \delta\text{: } \delta(q,0) = B \iff q = B\text{)} \\ &\iff w \text{ has an odd number of 1's.} && \text{(from 9)} \end{aligned}$$

Problem 4

(a) Show that every regular language is recognized by an ε-NFA where out of each state there is *no more than one* ε-transition and *no more than one* non-ε-transition (i.e., a transition on a symbol from the alphabet).

(b) Show that every regular language is recognized by an ε-NFA where out of each state there is *exactly one* ε-transition and *exactly one* non-ε-transition (i.e., a transition on a symbol from the alphabet). (A solution to this part is obviously also a solution to the previous part.)

Since proving b proves a, that is the only proof written.
First, we can use our favorite method to remove every ε-transitions. Let us look at the states with multiple alphabet transitions, which we'll call q.
Let $t_1, \ldots, t_k$ be the transitions of q, where $k \geq 2$. Then we can replace q with k new states, called $q_1, \ldots, q_k$, with a single ε-transition between sequential q_i's, and an alphabet transition of t_i from q_i to whatever the destination was before.
Now we have states with 1 or 0 ε-transitions and 1 or 0 alphabet transitions. For every state that doesn't have both types of transitions, make a new state that is rejecting and self-loops with both transitions, and then connect the first state to that new state by the transition type missing.
Now we created our ε-NFA with every state having exactly one ε-transition and one alphabet transition.

Problem 5 Do Exercise 3.1.1(b,c): Write regexes for the following languages:

b) The set of strings of 0's and 1's whose tenth symbol from the right end is 1.

c) The set of strings of 0's and 1's with at most one pair of consecutive 1's.

(b) The string must be at least 10 symbols long, but there can be anything before and anything after the 10^{th}-to-last symbol, so a regex for this language is

$$(0+1)^*1(0+1)^9.$$

(c) We can split any string in this language into three optional parts:

- The first part (if it exists) will have no consecutive 1's, and it needs to end in a 0 to make way for the second part.
- The second part is where the one pair of consecutive 1's can show up, but it will also allow for a singular 1 (in case the input is 1) or the empty string (in case there is no pair of consecutive 1's, or the input is empty).
- Finally, the third part (if it exists) will start with 0 to ensure there is no pair made with the second part, and this part will also contain no consecutive 1's.

So a regex for this language is

$$(0+10)^*(11+1+\varepsilon)(0+01)^*.$$

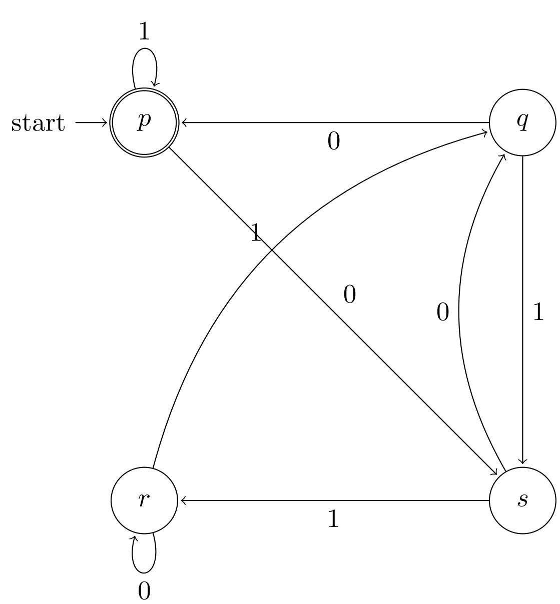
Problem 7,14 Write a regular expression for the language of strings over $\{a,b,c\}$ where no a appears after any b or c.

Since no a can occur after b or c, any string in this language will consist of a (possibly empty) string of a's, and then a string of b's and c's. So a regex for this language is

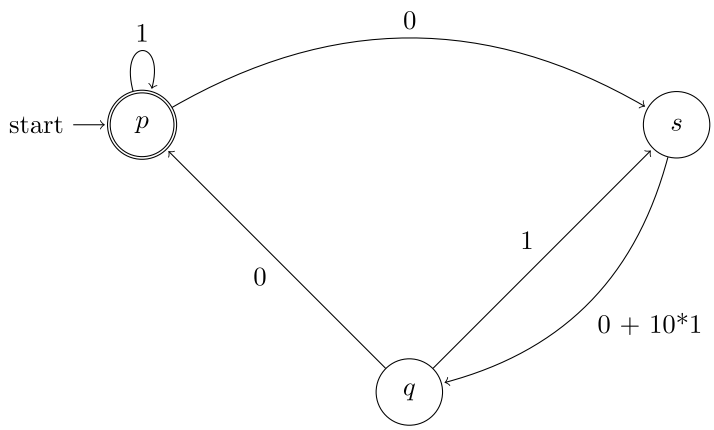
$$a^*(b+c)^*.$$

Problem 8 Do Exercise 3.2.3: Convert the following DFA to a regular expression, using the state-elimination technique of Section 3.2.2.

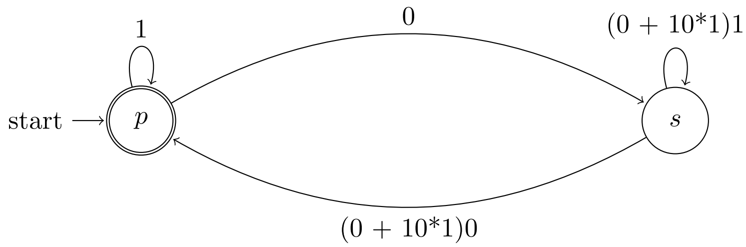
	0	1
$\rightarrow *p$	s	p
q	p	s
r	r	q
s	q	r



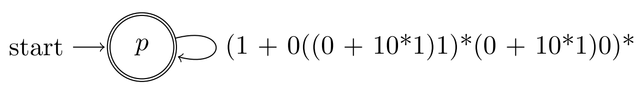
Since r has the fewest bypasses it is eliminated first.



We then eliminate q (q and s are tied for bypasses).

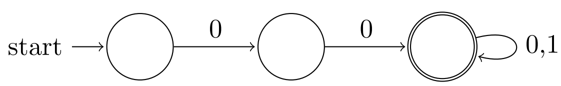


We finally eliminate s to only have p and produce the regex.



Problem 9 Do Exercise 3.2.4(c): Convert the following regex to an ε-NFA: $\mathbf{00}(\mathbf{0}+\mathbf{1})^*$.

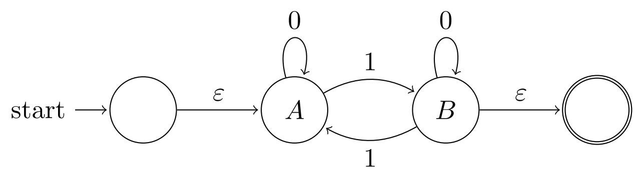
This is clear and doesn't need to be complicated by adding a billion ε moves. We add a state for the first 0, then another one for the second 0, then that last state is accepting with a loop for a 0 or 1 since it can be any number of either.



Problem 10 Recall the DFA D we constructed that accepts a binary string iff it has an odd number of 1's:

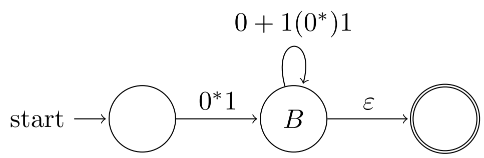
	0	1
$\to A$	A	B
$*B$	B	A

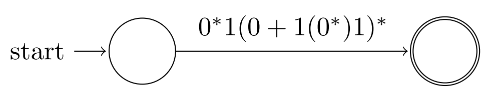
(a) Convert D into an equivalent clean ε-NFA using the clean-up procedure in class (add a new start state, a new final state, and some ε-transitions).



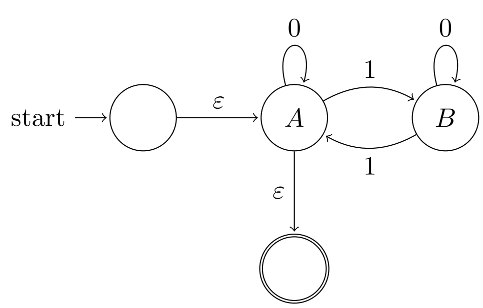
(b) Use the state elimination method to convert D to a regular expression. Eliminate state A first, then B.

The loop on A symbolizes an unspecified amount of 0's, and then needs a 1 to move closer to an accepting state, so replace A with edge of 0^*1.
Imagining we are at B and we don't take the ε move to the accepting state, we can go to either stay at B or move to state A and stay for any amount of time and move back to B. That thought can happen any number of times, so this is represented by a loop of $0 + 1(0^*)1$.

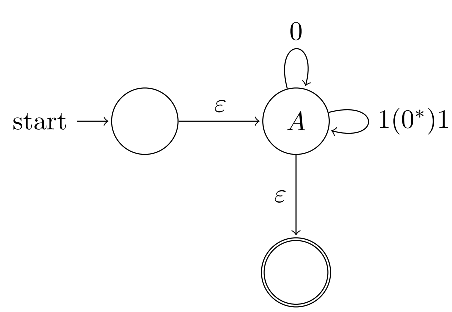




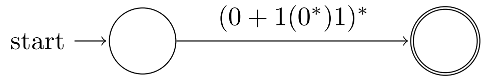
Problem 11 Same exercise as Problem 10, except make A the final state (so that D accepts a string iff it has an *even* number of 1's).



Let us remove state B first since it is not connected to the start and end states. The drawing of B already kind of looks like a loop, so it's easy to see that all of B stuff can be replaced by a loop of $1(0^*)1$.



So each loop on A can be combined with $+$, and since they are loops they represent it happening any number of times.



Problem 13 Draw the transition diagram of an ε-NFA equivalent to the regex $(a + bc)^*aa$. You may (but are not required to) contract ε-transitions provided it is safe to do so.