All students should read §3.3–3.6.

Distribute the following questions across the members of your group. You will share your solutions (and most importantly the *method* of your solutions) during the next lecture period. Divide up the questions so that **each** question has at least two solutions from different group members.

For all of these questions, you do not need to venture farther than the **regular expression language** described in the text for answers (namely: a, λ, $a|b$, a^+, a^k, and (of course) $a*$). You don't need "look backs", named groups, or other notions from any other regex language.

Questions for Review

For **questions 1, 2, and 4**, provide an algorithm that proves the assertion.

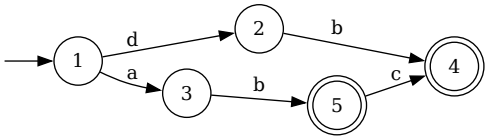
1. Page 110, question 18[1]

Solution: Given e a regular expression, show that the strings **not** accepted by e, $\bar{e}$, is a regular set.

IDEA: If we can show that a DFA exists that accepts $\bar{e}$, then $\bar{e}$ is a regular set.

Proof: Let D be the DFA with n states $i = 1,\ldots,n$ equivalent to e (we know there is equivalency between regular expressions and DFAs).

Suppose $e = (abc|(a|d)b), \Sigma = \{a,b,c,d\}$; the equivalent DFA would be

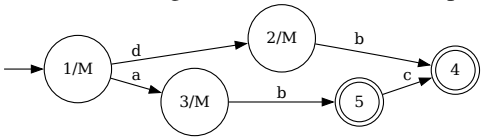


Let C_i be all the characters in Σ **without** an emanating edge from state i in D. For e above:

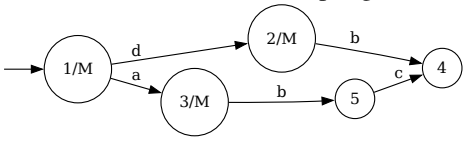
i	C_i
1	$\{b,c\}$
2	$\{a,c,d\}$
3	$\{a,c,d\}$
4	Σ
5	$\{a,b,d\}$

We use a proof by construction:

1. Let $\overline{D}$ be a copy of D
2. Mark all the stages in $\overline{D}$ that are non-accepting states.



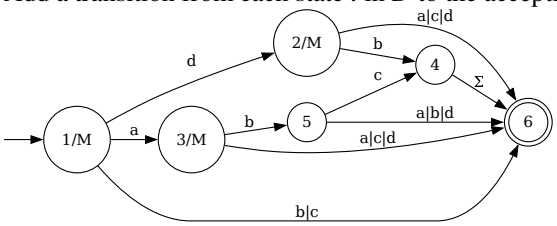
3. Make all states in $\overline{D}$ non-accepting states.



[1] Too bad $Not(e)$ doesn't count as an algorithm.

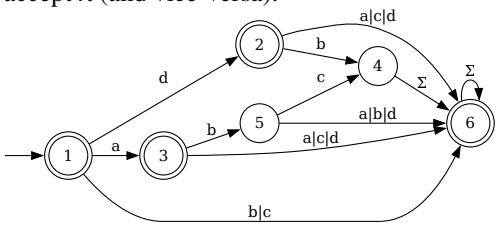
4. Add a single new accepting state $n+1$ to $\overline{D}$.

5. Add a transition from each state i in $\overline{D}$ to the accepting state of $\overline{D}$ for the characters in C_i.



6. Add a self-incident edge to the accepting state of $\overline{D}$ for Σ

7. Finally, make all the marked nodes accepting nodes. Note: if D does not accept λ, this will make $\overline{D}$ accept λ (and vice-versa).



The idea behind this construction is that we have "trapped" all the **sentences** ("words?", character sequences) that D will accept into non-accepting nodes (step 3) and provided an accepting path in $\overline{D}$ for any sentence D would not match.

Lastly, toggling the "accepting" attribute of each state in the DFA doesn't work. Consider the REs λ or $\lambda|a|ab$, you'll be left with a modified DFA without any accepting states!

2. Page 110, question 19; **Hint:** the world does not revolve around DFAs.

Solution: By definition a RE is a concatenation of (sub) REs which may themselves be a concatenation (sequence), an alternation ($|$), or a Kleene closure ($*$).

Alternation and Kleene closure are invariant with respect to "direction" or order; all we have to do is worry about the RE sequences.

Given RE e:

1. rewrite any high level operators (a^+, a^k) using only the four fundamental operators (sequence, alternation, Kleene closure, grouping)
2. in a recursive manner reverse the element order of all sequences (being sure to maintain the correct binding of Kleene operators to their arguments).

Violà.

3. Page 111, question 25

Solution: A RE $Seq(x,y)$ matching **non-zero length strings** of alternating characters x and y:

$$Seq(x,y) = (x(yx)*(y|\lambda))|(y(xy)*(x|\lambda)$$

Within each alternated pattern, the first character guarantees a non-zero length string, the Kleene closure permits only the appropriate character pattern given the first character, and the final group permits the pattern to end with either the starting character (in the Kleene closure) or the other character.

Now use $Seq(x,y)$ to construct S the regular set of non-zero length strings from $\Sigma = \{a,b,c\}$ beginning with a and having no two letters in a row.

IDEA: We define β to an RE that matches a string **of length greater than one** beginning with a, followed only by an alternating pattern of bs and cs (which means **not ending in** a).

$$\beta = (a(Seq(b,c)|Seq(c,b)))^+$$

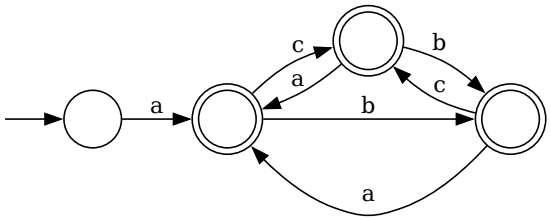
Recall $Seq(b,c)$ matches {"b","bc","bcb",...}. So $(a\,Seq(b,c))^+$ matches {"ab","abc","$abcb$",...}. Also, $Seq(b,c)$ may end with either a b or c — so avoid matching two sequential identical characters, we can never have a $Seq(x,y)$ immediately followed by any other $Seq(x,y)$ **or** $Seq(y,x)$. The construction of β avoids this by always beginning with a.

Put it altogether

$$\begin{aligned} S &= a|\beta|\beta a \qquad (1)\\ &= a|\beta(a|\lambda) \qquad (2)\end{aligned}$$

Given you have $Seq(x,y)$ and β defined, equation (1) on a quiz or exam would earn full credit.

It turns out you could probably reason out a correct DFA for the described pattern:



and it seems counter-intuitive that such a long RE would produce such a simple DFA

$$a|(a(((b(cb)*(c|))|(c(bc)*(b|)))|(((c(bc)*(b|))|(b(cb)*(c|))))))^+(a|)$$

Between the two is a `very large NFA` (also posted on the schedule page) which will be one of the (possible) outputs of the `WRECK` project this semester.

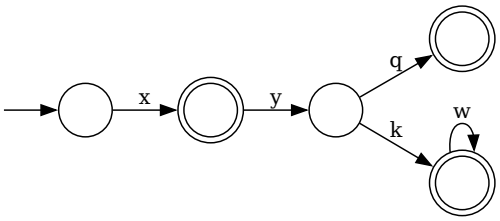
4. Page 111, question 26; **Warning:** In the question, $AllButLast(a^+b) = a^+$ should be interpreted as: "Applying $AllButLast$ to the regular set generated by a^+b would generate a set of strings that would all be matched by the RE a^+. It **does not mean** $AllButLast$ applied to the sequence of characters a, superscript $+$, and b would yield a^+ — which, of course it **would** if that is what the author meant.

Solution: **IDEA:** remove each edge to an accepting state for DFA R, make the "source" state of these edges accepting states. Why? Because if we can create a DFA for $AllButLast(R)$, then $AllButLast(R)$ defines a regular set.

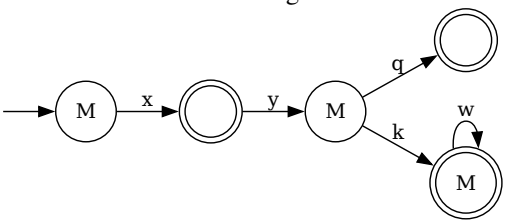
BEWARE! Accepting states with **self-incident** edges.

Suppose $R = x(\lambda|y(q|kw*))$,

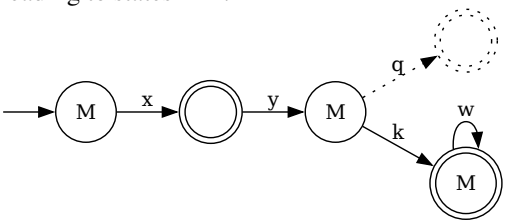
Let D be the DFA of R, which will turn into a DFA of $AllButLast(R)$:



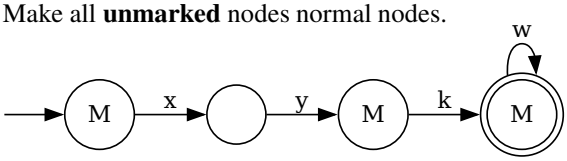
1. Mark all the states in D that have an edge leading to an accepting state in D — **including** accepting states with self-incident edges!



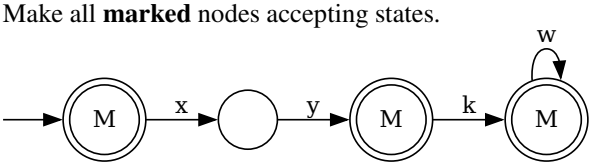
2. Let A be the set of all accepting states **without** emanating edges. Remove the states in A and all edges leading to states in A.



3. Make all **unmarked** nodes normal nodes.



4. Make all **marked** nodes accepting states.



Check:

R	$AllButLast(R)$
x	λ
xyq	xy
xyk	xy
$xykw$	xyk
$(m > 1)\ xykw^m$	$xykw^{m-1}$

5. (a) Find a reasonably good tutorial or short article on LEX (one you can read and understand); provide this to your group for their future benefit.

   (b) Page 111, question 22; **Hints:** "before the blank" and "very last character" refers to ASCII table ordering.

$x^{12,345}$ is a regular expression: 12,345 `x` characters in a row. The book doesn't mention this RE form in LEX, so it is a good thing you found a good tutorial in part a.

Solution: See `page111q22.l` on disk, run `make test-q22`.