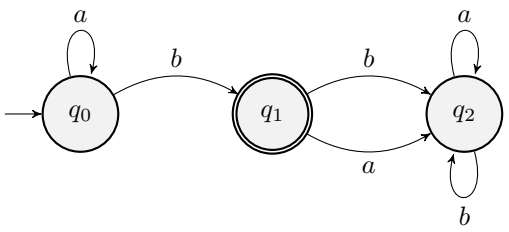
4. Regular Languages:

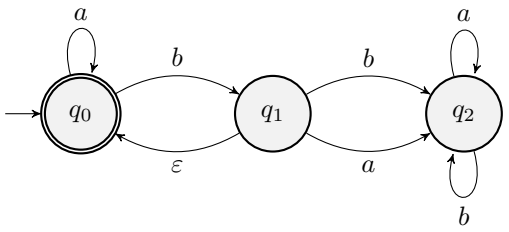
1. Why can't the automaton $Star(M)$ used in step (iv) of the proof of part (a) of Kleene's Theorem be constructed simply by taking M, making its start state the only accepting state and adding new ε-transitions back from each old accepting state to its start state?

   If we add new ε-transitions back from each old accepting state to the start state and make the start state accepting then the resulting automaton will accept a language which is a superset of the language accepted by the previous automaton (†). However it is not necessarily equal since it is possible for automatons to return to their start state without being accepted.

   Consider the minimal counterexample $M = a^*b$. Here is a diagram of the *DFA* M:



   If we add an epsilon transition from q_1 to q_0, make q_0 accepting and q_1 non-accepting then the NFA^ε formed will now also accept a^*. Consider specifically a. It is easy to see that a is accepted by this NFA^ε while it is not accepted in $Star(M)$. So this NFA^ε is not $Star(M)$.



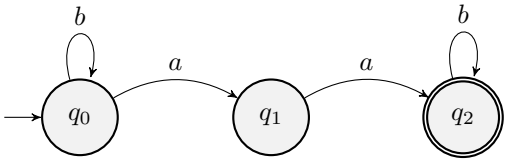
   (†) Assume that the previous automaton is in an accepting state.

   Consider the automaton formed by adding ε-transitions from old accepting states to the start state and making the start state the only accepting state.

   The input leaves the new automaton in one of the old accepting states. However since there is a ε-transition to an accepting state, the new automaton is now in an accepting state.

   So the language accepted by the new automaton is a superset of the language accepted by the old automaton.

2. Construct an NFA^ε M satisfying $L(M) = L((\epsilon|b)^*aab^*)$.



3. Show that any finite set of strings is a regular language.

   Let S be an arbitrary finite set of strings over the alphabet Σ.

By definition a string is a finite sequence of symbols in the alphabet Σ.

$$\forall s \in S.s \textit{ finite} \implies$$
$$\forall s \in S.\{s\} = L(s) \iff$$
$$\forall s \in S.\exists DFA\ d.\{s\} = L(d) \text{ using Kleene's Theorem}$$

Now consider the set of *DFA*'s D such that every $d \in D$ recognises a unique $s \in S$.

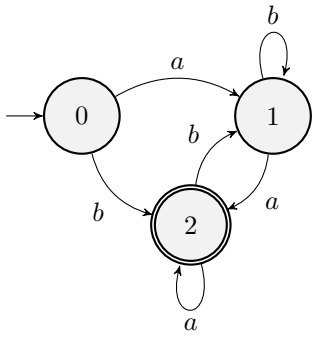
Consider now the NFA^ε which has a new start state and ε-transitions to the start state of every $d \in D$. This NFA^ε must accept all strings which are accepted by any $d \in D$. So this NFA^ε must accept all strings $s \in S$.

Since there exists a NFA^ε which accepts S; by Kleene's Theorem S is a regular language. Since S was arbitrary we can conclude that any finite set of strings is a regular language.

4. Use the construction given in the proof of part (b) of Kleene's Theorem to find a regular expression for the *DFA* M whose state set is $\{0, 1, 2\}$, whose start state is 0, whose only accepting state is 2, whose alphabet of input symbols is $\{a, b\}$, and whose next-state function is given by the following table.

δ	a	b
0	1	2
1	2	1
2	2	1

Here is a diagram of the *DFA* M:



Part (b) of Kleene's Theorem has four parts:

- R^k_{ij} is regular expression for all paths from i to j which do not pass through any vertices $> k$.
- R^0_{ij} is the union of all edges from i to j.
- $R^k_{ij} = R^{k-1}_{ik}(R^{k-1}_{kk})^* R^{k-1}_{kj}$
- R^n_{0n} is the language accepted by the automata.

i	j	k	R_{ij}^k	
0	0	-1		ε
0	1	-1		a
0	2	-1		b
1	0	-1		$\emptyset$
1	1	-1		$b\|\varepsilon$
1	2	-1		a
2	0	-1		$\emptyset$
2	1	-1		b
2	2	-1		$a\|\varepsilon$
0	0	0	$\varepsilon\varepsilon^*\varepsilon\|\varepsilon$	$= \varepsilon$
0	1	0	$\varepsilon\varepsilon^*a\|a$	$= a$
0	2	0	$\varepsilon\varepsilon^*b\|b$	b
1	0	0	$\emptyset\varepsilon^*\varepsilon\|\emptyset$	$= \emptyset$
1	1	0	$\emptyset\varepsilon^*a\|b\|\varepsilon$	$= b\|\varepsilon$
1	2	0	$\emptyset\varepsilon^*b\|a$	$= a$
2	0	0	$\emptyset\varepsilon^*\varepsilon\|\emptyset$	$= \emptyset$
2	1	0	$\emptyset\varepsilon^*a\|b$	$= b$
2	2	0	$\emptyset\varepsilon^*\varepsilon\|a\|\varepsilon$	$= a\|\varepsilon$
0	0	1	$a(b\|\varepsilon)^*\emptyset\|\varepsilon$	$= \varepsilon$
0	1	1	$a(b\|\varepsilon)^*(b\|\varepsilon)\|a$	$= ab^*$
0	2	1	$a(b\|\varepsilon)^*a\|b$	$= ab^*a\|b$
1	0	1	$(b\|\varepsilon)(b\|\varepsilon)^*\emptyset\|\emptyset$	$= \emptyset$
1	1	1	$(b\|\varepsilon)(b\|\varepsilon)^*(b\|\varepsilon)$	$= b^*$
1	2	1	$(b\|\varepsilon)(b\|\varepsilon)^*a\|a$	$= b^*a$
2	0	1	$b(b\|\varepsilon)^*\emptyset\|\emptyset$	$= \emptyset$
2	1	1	$b(b\|\varepsilon)^*(b\|\varepsilon)\|b$	$= bb^*$
2	2	1	$b(b\|\varepsilon)^*a\|a\|\varepsilon$	$= b^*a\|\varepsilon$
0	2	2	$(ab^*a\|b)(b^*a\|\varepsilon)^*(b^*a\|\varepsilon)\|(ab^*a\|b)$	$= (ab^*a\|b)(b^*a)^*$

So the regular expression which is equivalent to the *DFA* is:

$$(ab^*a|b)(b^*a)^*$$

5. If $M = (Q, \Sigma, \Delta, s, F)$ is an *NFA* , let $Not(M)$ be the *NFA* $(Q, \Sigma, \Delta, s, Q/F)$ obtained from M by interchaining the role of accepting and non-accepting states. Give an example of an alphabet Σ and an *NFA* M with the set of input symbols Σ, such that $\{u \in \Sigma^* \mid u \notin L(M)\}$ is *not* the same set as $L(Not(M))$.

   Consider the *NFA* M with input alphabet $\Sigma = \{a\}$ which accepts only ε.

   Here is a diagram of M:

   Note that M does not accept the string a – or more generally any string of the form aa^*, however my proof involves only a.

   Consider now $Not(M)$:

   $Not(M)$ has no accepting states and so accepts no strings, namely a.

So:

$$\begin{aligned} a \notin L(M) \wedge a \notin L(Not(M)) &\iff \\ a \in \{u \in \Sigma^* | u \notin L(M)\} \wedge a \notin a \in L(Not(M)) &\implies \\ \{u \in \Sigma^* | u \notin L(M)\} \neq L(Not(M)) &\text{ as required} \end{aligned}$$

6. Let $r = (a|b)^*ab(a|b)^*$. Find a complement for r over the alphabet $\{a, b\}$, i.e. a regular epression $\sim r$ over the alphabet $\{a, b\}$ satisfying $L(\sim r) = \{u \in \{a, b\}^* | u \notin L(r)\}$.

$$\sim r = b^*a^*$$

7. Given *DFA*s $M_i = (Q_i, \Sigma, \delta_i, s_i, F_i)$ for $i = 1, 2$, let $And(M_1, M_2)$ be the *DFA* $(Q_1 \times Q_2, \Sigma, \delta, (s_1, s_2), F_1 \times F_2)$ where $\delta : (Q_1 \times Q_2) \times \Sigma \longrightarrow (Q_1 \times Q_2)$ is given by:

$$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$$

for all $q_1 \in Q_1, q_2 \in Q_2$ and $a \in \Sigma$. Show that $L(And(M_1, M_2))$ is the intersection of $L(M_1)$ and $L(M_2)$.

I will show by induction that the state of $And(M_1, M_2)$ after a string of length n, s_n is equal to (q_1, q_2) where q_1 is M_1's state and q_2 is M_2's state.

If $s = \varepsilon$:

Then all *DFA*'s are in their start states. So the state of $And(M_1, M_2)$ is (s_1, s_2), the state of M_1 is s_1 and the state of M_2 is s_2. So the statement holds for $s = \varepsilon$.

Assume that the statement holds for some string s_k.

Let s_{k+1} be a string such that $s = s_k v$ where $v \in \Sigma$. So the state of $And(M_1, M_2)$ after string s is $\delta((q_1, q_2), v) = (\delta_1(q_1, v), \delta_2(q_2, v))$, the state of M_1 after string s is $\delta_1(q_1, v)$ and the state of M_2 after string s is $\delta_2(q_2, v)$. So if the statement holds for some arbitrary string s_k of length k then it also holds for all strings s_{k+1} of length $k + 1$.

Since the statement holds for $s = \varepsilon$ and the statement holding for a string of length k implies that it also works for a string of length $k + 1$.

So this implies that $\forall n \in \mathbb{N}$ after input s_n, the state of $And(M_1, M_2)$ is equal to (q_1, q_2) where q_1 is the state of M_1 and q_2 is the state of M_2.

The accepting states of $And(M_1, M_2)$ are (f_1, f_2) where $f_1 \in F_1, f_2 \in F_2$. By the proof above this means that $And(M_1, M_2)$ is in an accepting state if and only if both M_1 and M_2 are in accepting states.

Hence $L(And(M_1, M_2))$ is the intersection of $L(M_1)$ and $L(M_2)$.

5. The Pumping Lemma:

1. Consider

$$L \triangleq \{c^m a^n b^n | m \geq 1 \wedge n \geq 0\} \cup \{a^m b^n | m, n \geq 0\}$$

The notes show that this language has the pumping lemma property. Show that there is no *DFA* M which accepts L. [*Hint*: argue by contradiction. If there were such an M, consider the *DFA* M_0 with the same states as M, with alphabet of input symbols just consisting of a and b, with transitions all those of M which are labelled by a or b, with start state $\delta_M(s_M, c)$ (where s_M is the start state of M), and with the same accepting states as M. Show that the language accepted by M' has to be $\{a^n b^n | n \geq 0\}$ and deduce that no such M can exist].

Assume L is a regular language.

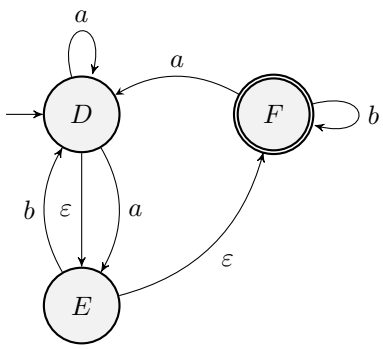
This implies that there is a *DFA* D which recognises L.
Since D recognises L, it must recognise $\{c^m a^n b^n\}$ (since this is a subset of L).
Now consider a *NFA* which contains only the a, b transitions in D.
Since the *DFA* recognises $\{c^m a^n b^n\}$, without c transitions the *NFA* will recognise $a^n b^n$.
However by the pumping lemma, $a^n b^n$ is not a regular language. By Kleene's Theorem, this is equivalent to saying there is no *NFA* which recognises $a^n b^n$.
This is a contradiction. So our original assumption that L is a regular language must be wrong.

So L is not a regular language.

1 2021 Paper 2 Question 10

https://www.cl.cam.ac.uk/teaching/exams/pastpapers/y2021p2q10.pdf

(a) Consider the following NFA^{ε} , whose input alphabet is $\{a, b, c\}$.



(i) For each of the two strings *abc* and *bba*, state whether the automaton accepts it, with justification

The automaton does not accept either *abc*.

For *abc* there is no transition for c and so it enters an undefined state which is not in the set of accepting states.

The automaton does accept *bba*.

It can ε-transition to from D to E, then transition to D with b. Repeat that again. Then transition to D using a. then ε-transition to E and ε-transition to F. This means that after *bba*, the automaton is in an accepting state and so *bba* is accepted by the automaton.

(ii) Using the subset construction, produce the full unoptimized state transition table of an equivalent *DFA* , *listing its states in lexicographic order* (important!) and indicating the starting and accepting states.

Q	a	b	c
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
{D}	{D, E, F}	{D, E, F}	$\emptyset$
{D, E}	{D, E, F}	{D, E, F}	$\emptyset$
{D, E, F}	{D, E, F}	{D, E, F}	$\emptyset$
{D, F}	{D, E, F}	{D, E, F}	$\emptyset$
{E}	{D, E, F}	{D, E, F}	$\emptyset$
{E, F}	{D, E, F}	{D, E, F}	$\emptyset$
{F}	{D, E, F}	{F}	$\emptyset$

The starting state is $\{D, E, F\}$.

The accepting states are $\{D, E, F\}, \{D, F\}, \{E, F\}, \{F\}$

(iii) Give a regular expression, no longer than six symbols (metacharacters included), that describes the strings accepted by the automaton, together with an intuitive explanation for it. [*Hint*: part (a)(ii) helps.]

Looking at the table in (a)(ii), note that the *DFA* starts in an accepting state and that the *DFA* will be in an accepting state after a transition iff it was in an accepting state before the transition and is passed either a or b.

From this, I gather that any sequence of a's and b's will be accepted. The regular expression for this is given below:

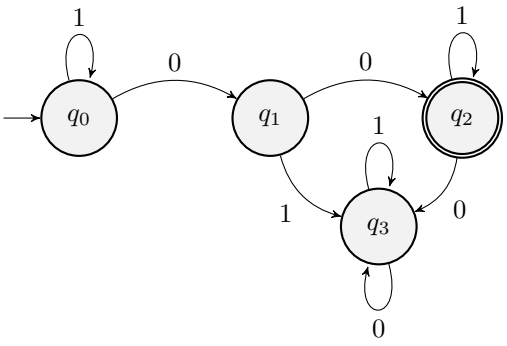
$$(a|b)^*$$

So any string of the form $(a|b)^*$ will be accepted by the automaton..

(b) Consider language L_1 of strings over alphabet $\{0, 1\}$, defined inductively as follows

$$\frac{}{00} \qquad \frac{w}{1w} \qquad \frac{w}{w1}$$

(i) Draw the diagram of a *DFA* that recognises L_1 in no more than four states.



(ii) Considering the words in L_1 as unsigned binary numerals, let language L_2 of strings over $\{0, 1\}$ be the set of all and only the binary numerals obtained by adding 1 to any numeral in L_1 and removing any leading zeros.
NB: "adding" here means arithmetic addition, not string concatenation.
Produce a regular expression no longer than 11 symbols that recognizes L_2, with a clear and convincing explanation of how you derived it.

$$L_2 = L((\varepsilon|1^*10)10^*)$$

- By inspection: L_1 is the set of strings which are of the form 1^*001^*.
- Adding one to a string $s \in L_1$ gives a string of the form 1^*010^*.
- If the number of ones preceding 00 was zero there will be a single leading zero to remove.

  This case gives strings of the form 10^*

  If the number of ones preceding 00 was nonzero then there are no leading zeros to remove.

  This case gives strings of the form 1^*1010^*.
- To form the regular expression which accepts the language L_2 we must take the union of these two regular expressions:

$$10^*|1^*1010^* = (\varepsilon|1^*10)10^*$$

So the regular expression which accepts the language L_2 is $(\varepsilon|1^*10)10^*$.