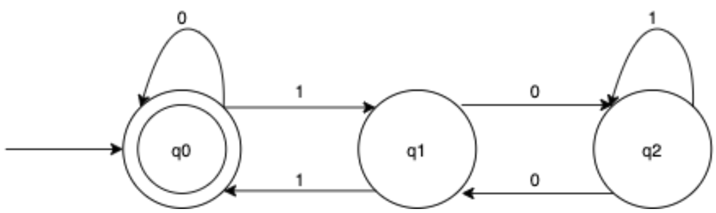
Problem 1: Regular expressions

A) ‘w’ does not end in ‘ba'
ε + a + b + (a + b)*(aa + bb + ab)

B) $w = \alpha \circ \beta$ and α has an even number of 1′s and β has an even number of 0′s
0*(10*10*)*1*(01*01*)*

Problem 2:

1. Construct a DFA to recognize the language D_3



2. The language D_3 accepts precisely those binary strings which when interpreted as numbers are exactly divisible by 3. Above figure presents a DFA for this language. The existence of a DFA for the language establishes its regularity.

   Identify that the DFA must have 3 states, one state to denote strings that are exactly divisible by 3, one state to denote strings that result in a remainder of 1 when divided by 3 and, another state to denote strings that result in a remainder of 2 when divided by 3.

   Consider the set of all strings 0s and 1s. Divide these strings into three sets

   E_0: all strings s.t. the number of 0s is a multiple of three
   E_1: all strings s.t. the number of 0s is one more than a multiple of three
   E_2: all strings s.t. the number of 0s is two more than a multiple of three

   Every string of 0s and 1s is in one of these three sets.

   If x and y are strings of 0s and 1s, then the concatenation xy is in D_3:
   If x is in E_0 then xy is in D_3 iff y is in E_0
   If x is in E_1 then xy is in D_3 iff y is in E_2
   If x is in E_2 then xy is in D_3 iff y is in E_1

   Observe that appending a 0 to the right of a binary number causes its value as a number to double, whereas adding a 1 results in a number that is sum of 1 and twice the original value.

$$If\ p \cong 0\ mod\ 3\ then\ 2p \cong 0\ mod\ 3\ and\ 2p + 1 \cong 1\ mod\ 3$$
$$If\ p \cong 1\ mod\ 3\ then\ 2p \cong 2\ mod\ 3\ and\ 2p + 1 \cong 0\ mod\ 3$$
$$If\ p \cong 2\ mod\ 3\ then\ 2p \cong 1\ mod\ 3\ and\ 2p + 1 \cong 2\ mod\ 3$$

Therefore there is no distinguishing extension for any two strings in the same one of three sets, so there are at most three equivalence classes. A finite number of equivalence classes means the language is regular.

Problem 3:

$L^R = \{w_1, \ldots, w_k | w_k, \ldots, w_1 \ \epsilon \ L, w_i \ \epsilon \ \Sigma\}$

If L is a regular language then so is L^R.

Proof 1

If L is recognized by an DFA, then L^R is recognized by DFA reading from right to left. Assume L is a regular language. Let M be a DFA that recognizes L. If M accepts w, then w describes a directed path in M from start to accept state. Try to define M^R as M with the arrows reversed. Turn start state into a final state. Turn final states into a start state.

Proof 2

Assume L is defined by a regular expression E. We show that there is another regular expression $E^R \ such \ that \ L(E^R) = (L(E))^R$. That is the language of E^R is the reversal of the language of E.

Basis: If E is a symbol a, ε, or ϕ, then $E^R = E$.

Induction:

1. E = F + G then $E^R = F^R + G^R$

   The reversal of the union of two languages is obtained by computing the reversal of the two languages and taking the union of these languages.

2. E = FG then $E^R = G^R F^R$

   We reverse the order of the two languages as well as reversing the languages themselves. In general if a word $w \in L(E)$ is the concatenation of $w_1 \in L(F)$ and $w_2 \in L(G) \ then \ w^R = w_2^R w_1^R$

3. E = F* then $E^R = (F^R)*$

   Any string $w \in L(E)$ can be written as $w_1 w_2 \ldots w_n$ where each w_i is in $L(F)$ but $w^R = w_n^R \ldots w_2^R w_1^R$ and each w_i^R is in $L(E^R)$ so w^R is in $L((F^R)*)$.

   Conversely any string in $L((F^R)*)$ is of the form $w_1 w_2 \ldots w_n$ where each w_i is the reversal of a string in L(F). The reversal of this string is in L(F*) which is in L(E).

   We have shown that a string is in L(E) iff its reversal is in $L((F^R)*)$

dtrived5@stevens.edu

Problem 4: DFA minimization

Transition table:

δ	A	B
1	2	3
2	5	4
3	4	9
4	6	8
5	2	6
6	4	7
7	8	6
8	7	4
9	8	3

Minimization table:

Round 1

	1	2	3	4	5	6	7	8	9
1				X					
2				X					
3				X					
4					X	X	X	X	X
5									
6									
7									
8									
9									

Round 2

	1	2	3	4	5	6	7	8	9
1		X	X	X		X		X	
2				X					
3				X					

	1	2	3	4	5	6	7	8	9
4					X	X	X	X	X
5									
6									
7									
8									
9									

Round 3

	1	2	3	4	5	6	7	8	9
1		X	X	X		X		X	
2			X	X	X	X	X		X
3				X					
4					X	X	X	X	X
5									
6									
7									
8									
9									

Round 4

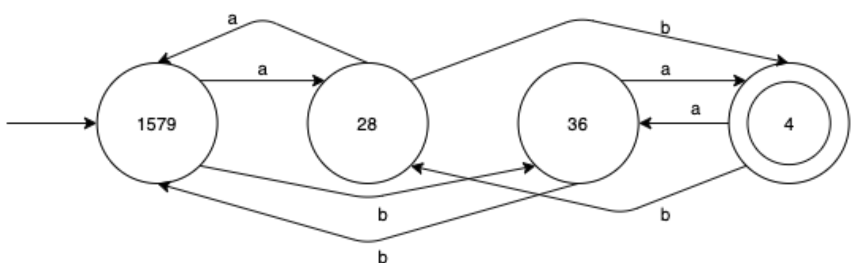
	1	2	3	4	5	6	7	8	9
1		X	X	X		X		X	
2			X	X	X	X	X		X
3				X	X		X	X	X
4					X	X	X	X	X
5									
6									
7									
8									
9									

Round 5

	1	2	3	4	5	6	7	8	9
1		X	X	X		X		X	
2			X	X	X	X	X		X
3				X	X		X	X	X
4					X	X	X	X	X
5						X		X	
6							X	X	X
7								X	
8									X
9									

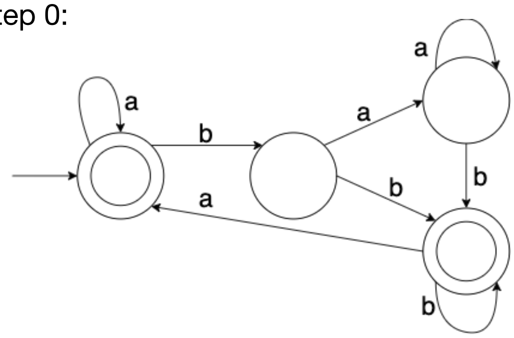
Pair of states to be merged are (1,5), (1,7), (1,9), (2,8), (3,6), (5,7), (5,9) and (7,9)

Minimized DFA:

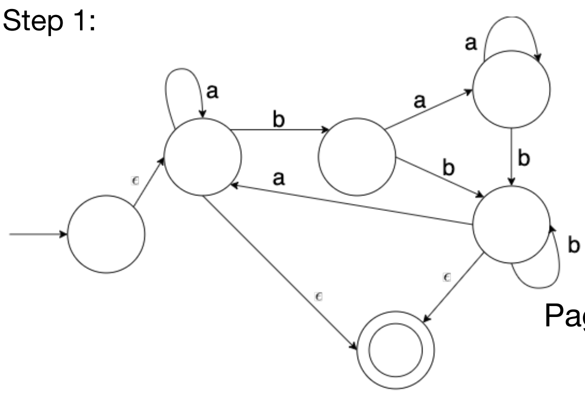


Problem 5:

Step 0:

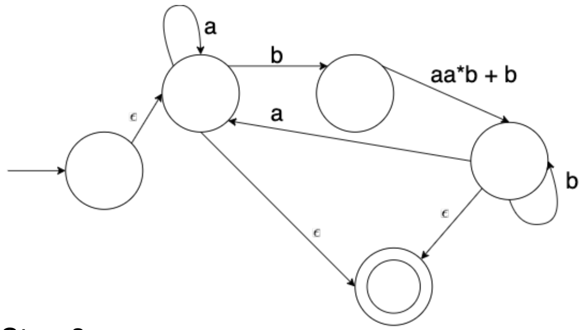


Step 1:

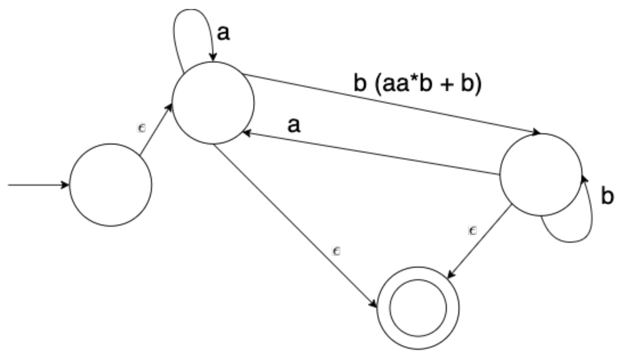


dtrived5@stevens.edu

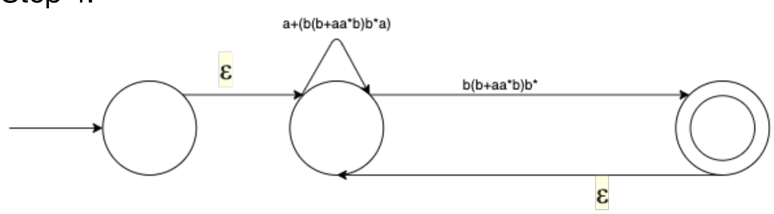
Step 2:



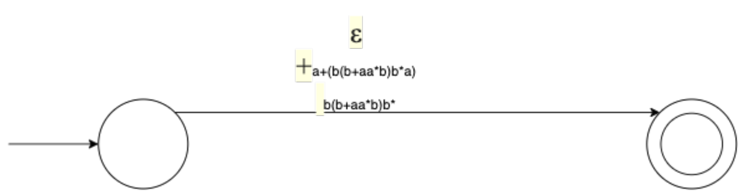
Step 3:



Step 4:



Step 5:



Regular expressions: $\epsilon + (a + (b(b + a^+b)b^*a))^*b(b + a^+b)b^*$

Problem 6:

L = $\{0^{2^i} : i \geq 0\}$

We start by proving that all regular languages have a pumping property (i.e. prove the pumping lemma). Then to show that language L is not regular, we show that L does not have the pumping property.

1. L regular implies L has a pumping property
2. L not having pumping property implies L is not regular

Suppose a DFA M accepts L. Assume L is regular. Let m be the number of states in M.

By the pumping lemma there exists a p such that every s ϵ L such that |s| $\geq$ p can be represented as xyz with |y| > 0 and |xy| $\leq$ p.

Choose s = 0^{2^i}, where 2^i > m

M must repeat states reading first m 0's.

If M accepts 0^{2^i}, then M accepts a string with 1 to m more 0's.

Since the length of xy cannot exceed p, y must be of the form 0^k for some 0 < k $\leq$ p.

We have $2^p < 2^p + k \leq 2^p + p < 2^{p+1}$.

However 2^i < 2^i + m < 2^{i+1}, i.e. the number of 0's won't be a power of 2.

Contradicts assumptions that M accepts L.