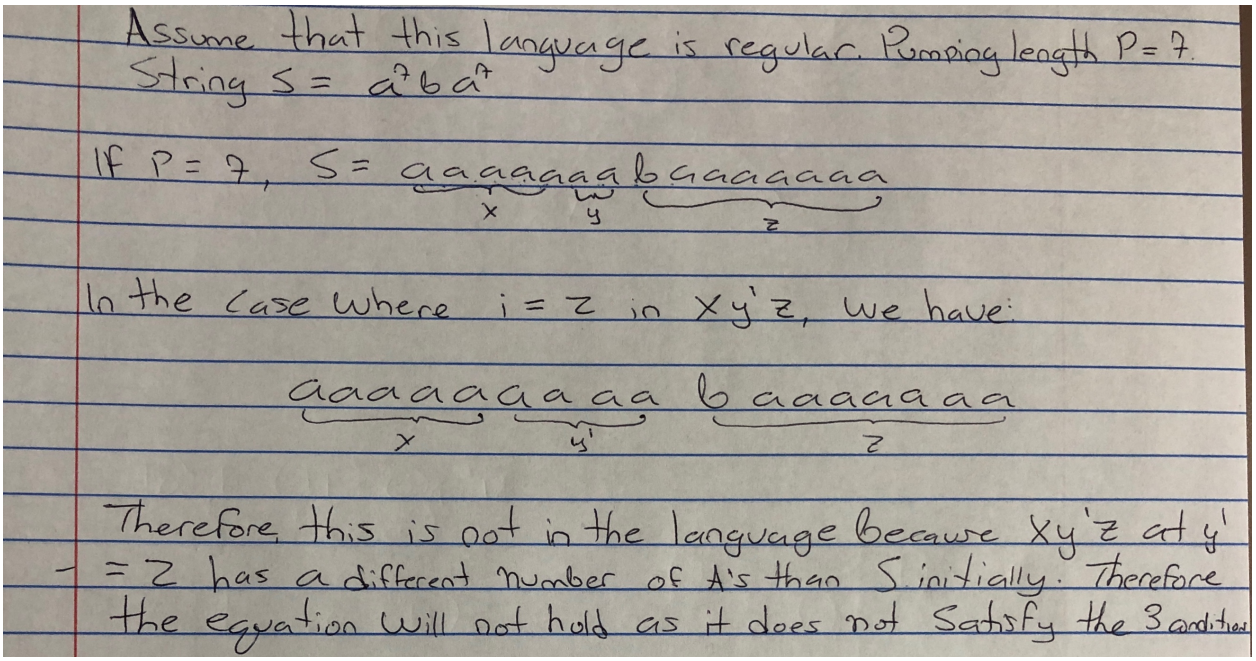
Firas Aboushamalah
250 920 750
CS 3331 Assignment #1
Lucian Ilie

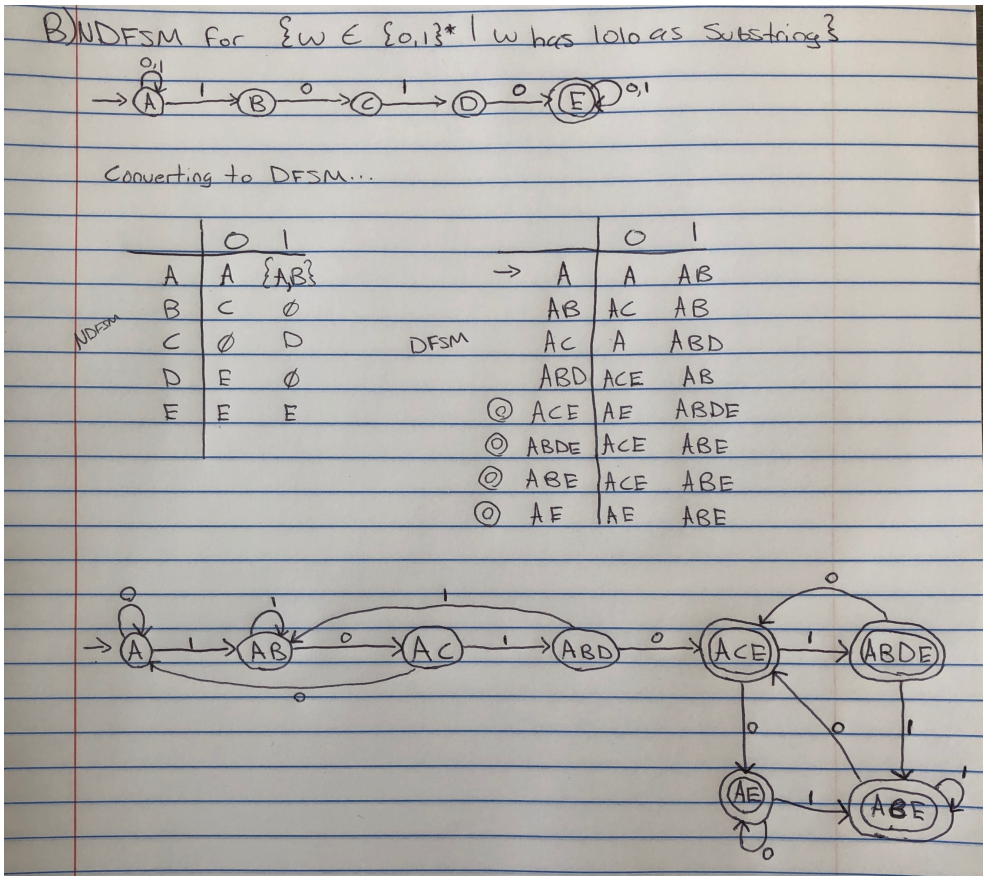
Problem 1)

A) This language is not regular.



B) **{w ∈ {0, 1} ∗ | w has 1010 as substring}.**

Making NDFSM and converting to DFSM

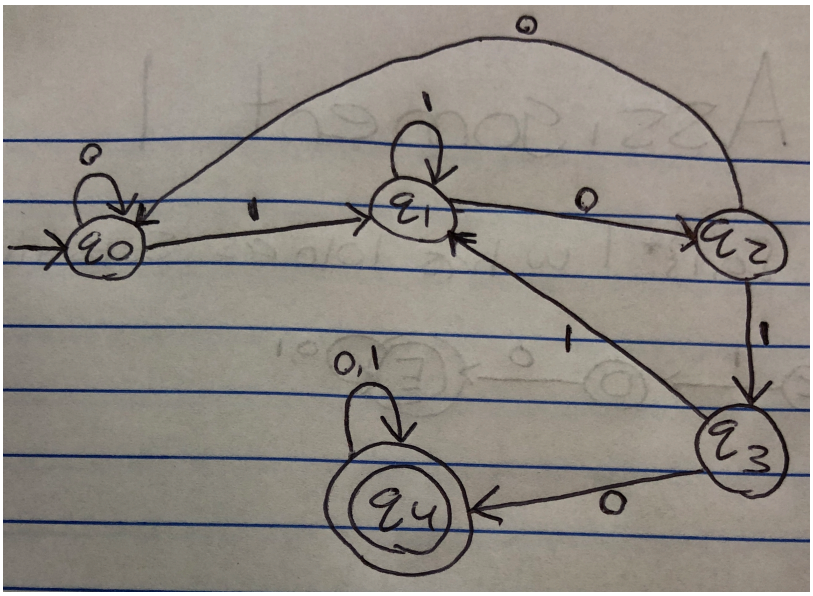


By minimizing this, we have the equivalence classes that are:

0 Equivalence = {A, AB, AC, ABD} {ACE, ABDE, ABE, AE}
1 Equivalence = {A, AB, AC} {ABD} {ACE, ABDE, ABE, AE}
2 Equivalence = {A, AB} {AC} {ABD} {ACE, ABDE, ABE, AE}
3 Equivalence = {A} {AB} {AC} {ABD} {ACE, ABDE, ABE, AE}

This is the final equivalence which means that we will have 5 total states and the fifth state will be accepting. We will call them {A} = q0, {AB} = q1, {AC} = q2, {ABD} = q3, and {ACE, ABDE, ABE, AE} = q4.

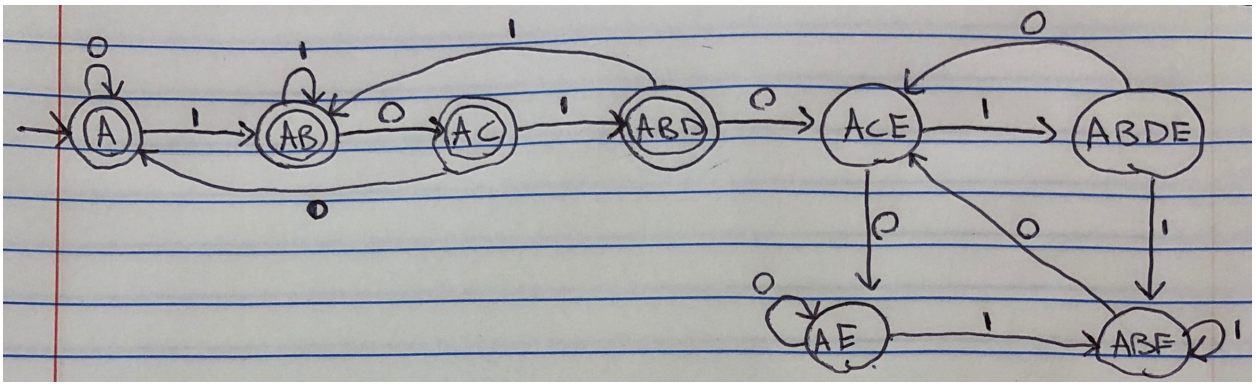
Therefore, we have as the minimal DFSM:



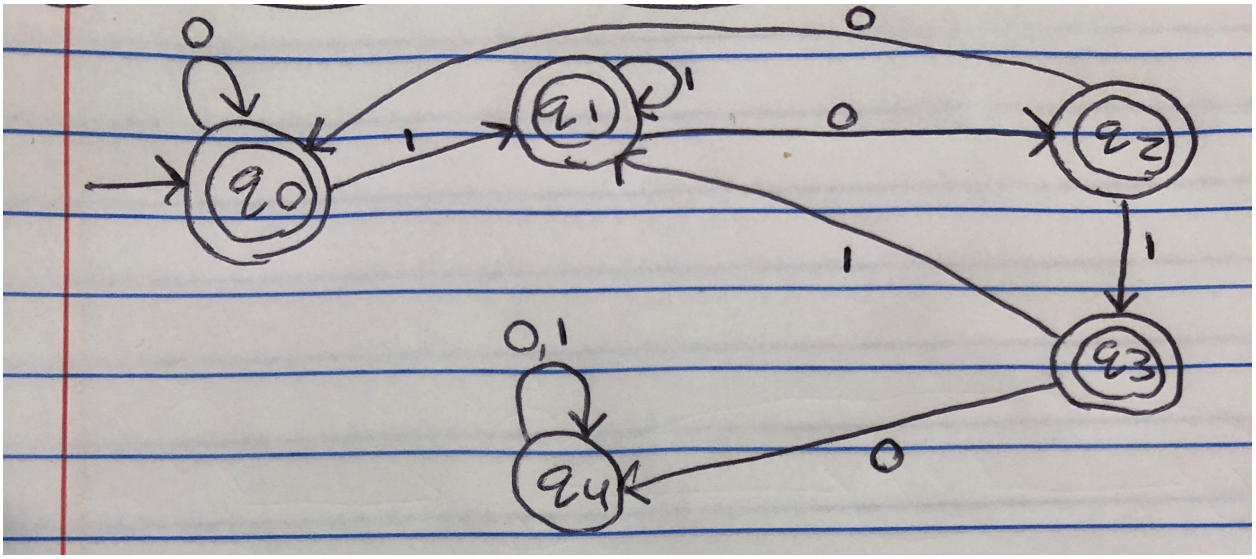
Regular Expression: (1|0)* 1010 (1|0)*

C) {w ∈ {0, 1} ∗ | w does not have 1010 as substring}.

The simplest way to do this is to solve for "w DOES have 1010 as a substring" and then reverse the non-accepting states to accepting and accepting states to non-accepting. Because we already did this, the DFSM for this is:



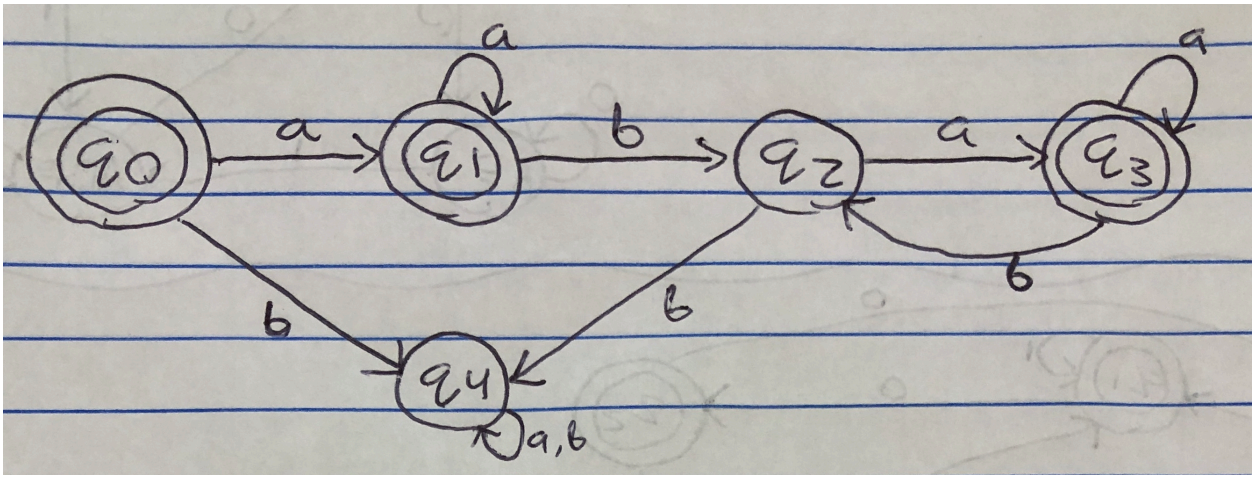
Meanwhile the minimal DFSM is (just reversing states):



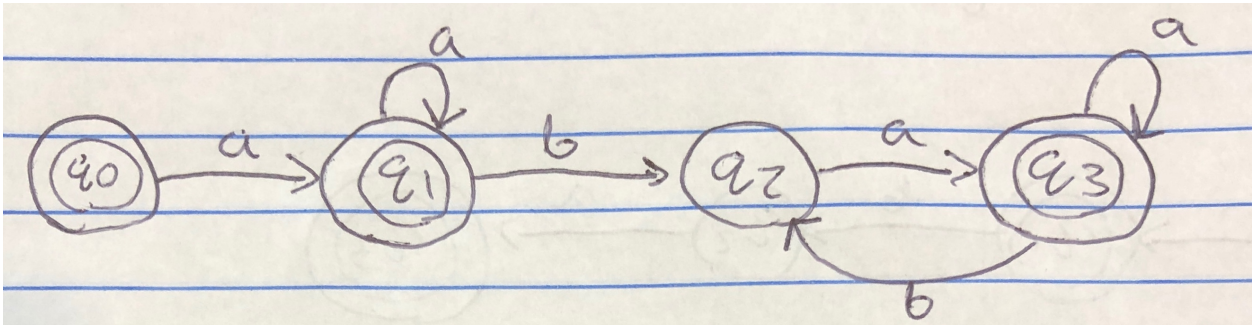
Regular Expression: $(0 + 11^*00 + 11^*01(11^*01)^*11^*00)^*(\text{eps} + 11^* + 11^*0 + 11^*01(11^*01)^*(\text{eps} + 11^* + 11^*0))$

D) {w ∈ {a, b} ∗ | every b in w is immediately preceded and followed by a}.

The direct DFSM is:



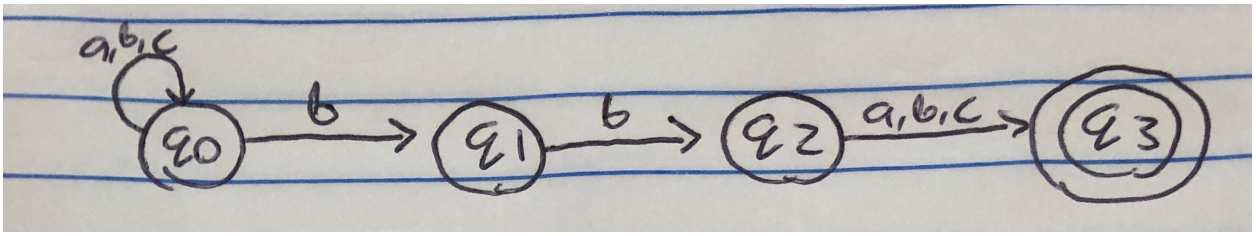
The minimal DFSM is made by removing the trap state q4, otherwise this is already minimized:



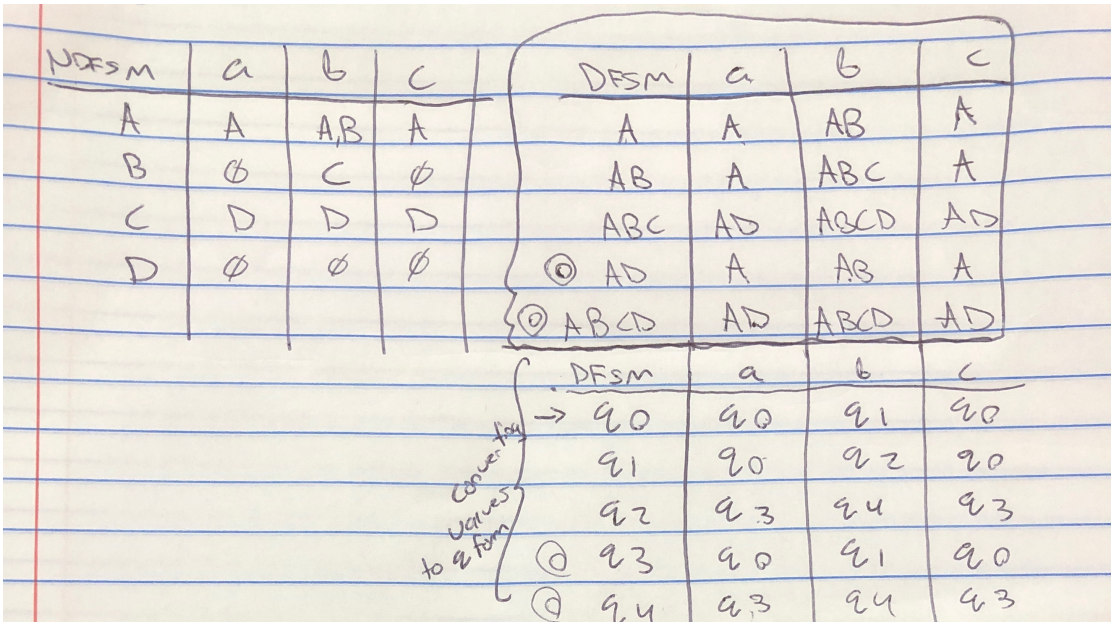
Regular Expression: (a*ab)(a*ab)*aa* + a*

E) {w ∈ {a, b, c} ∗ | the third and second from the last characters are b's}.

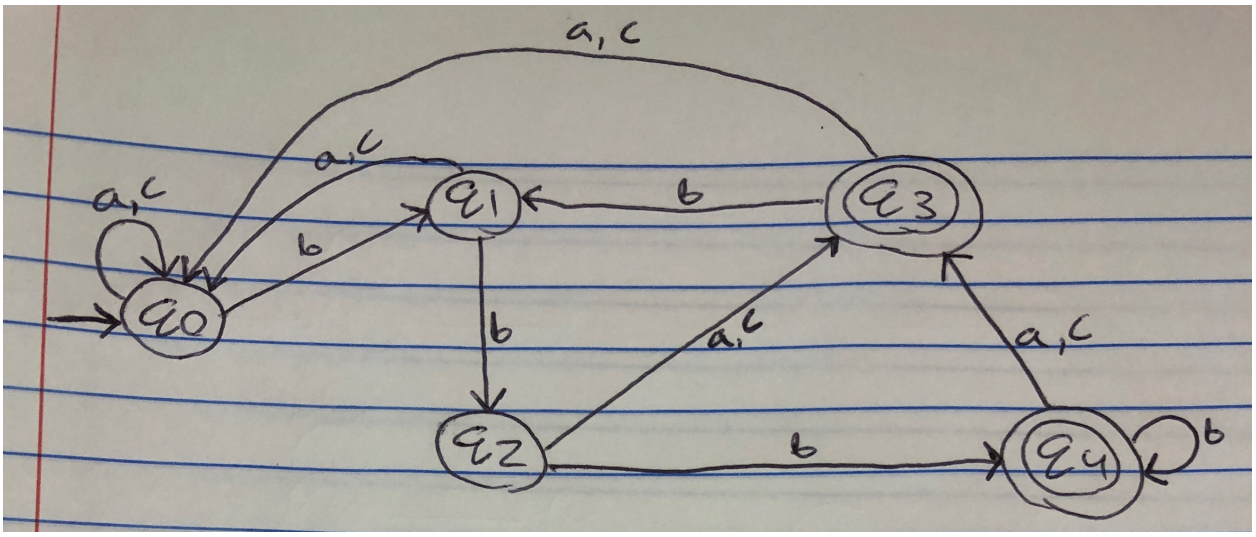
The NDFSM for this language is:



To create the DFSM, we write out the state transition table (**NOTE**: in order to help me understand easier, I convert the state names from q0, q1.. etc. to A, B, C or q1+q2 = AB.. I convert them back though as can be seen in the lower table)



After mapping these values (lower right table), we get the DFSM:



To minimize this, we write out the equivalence tables:

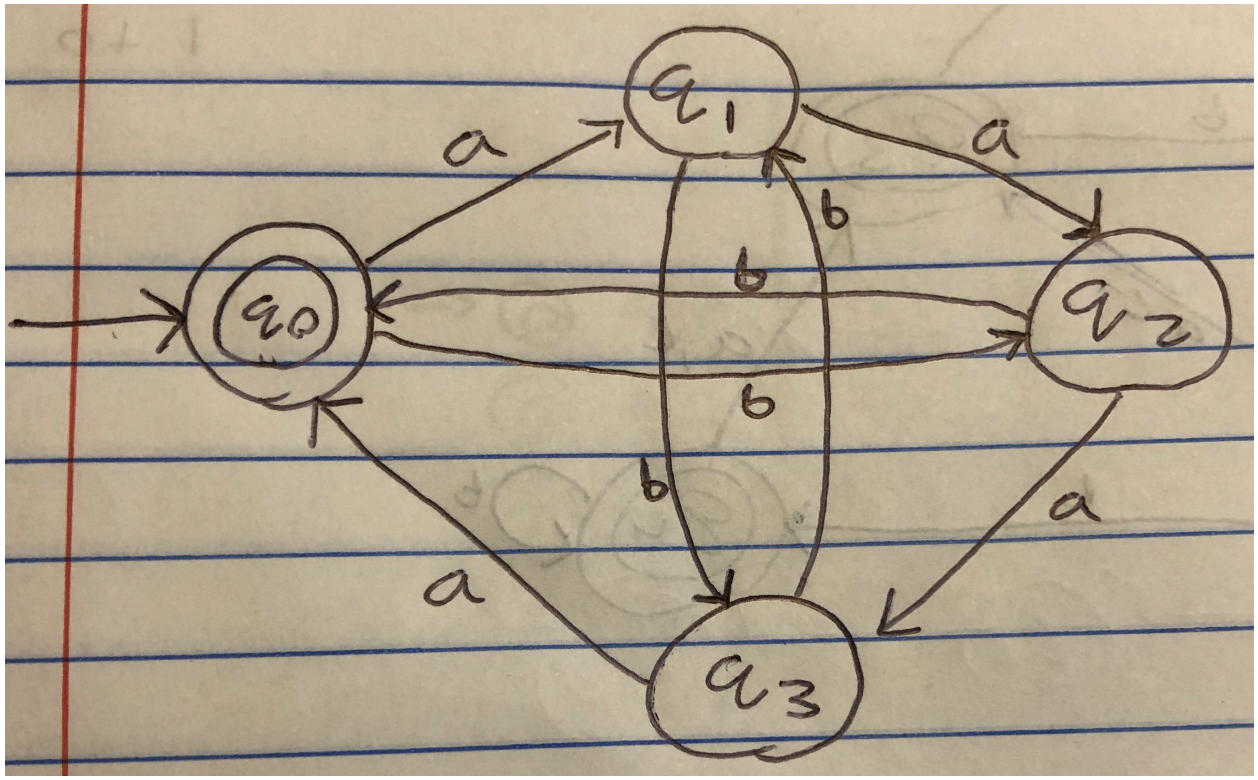
0 Equivalence: {q0, q1, q2} {q3, q4}
1 Equivalence: {q0, q1} {q2} {q3, q4}
2 Equivalence: {q0} {q1} {q2} {q3, q4}
3 Equivalence: {q0} {q1} {q2} {q3} {q4} where q0 = start and q3/q4 = accepting states

Therefore, the DFSM is already minimized in the second state diagram because no splitting.

Regular Expression: (a|b|c)*bb(a|b|c)

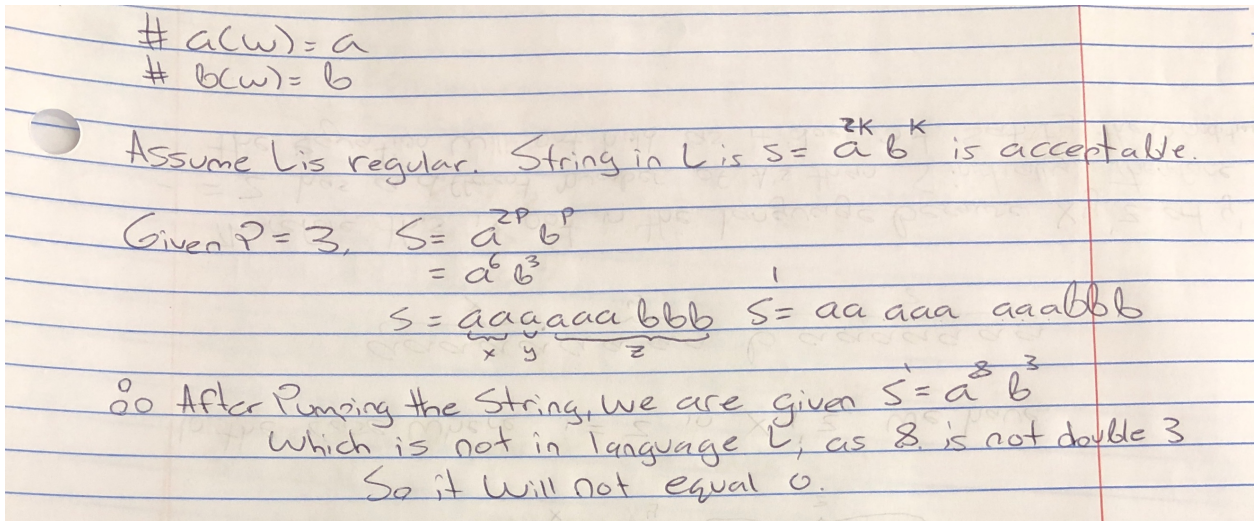
F) {w ∈ {a, b} ∗ | (#a(w) + 2#b(w)) ≡ 0 (mod 4)}.

The following is a DFSM (not NDFSM) because there is only one possible state for each value and, it is already minimized because there is no unreachable or redundant states.



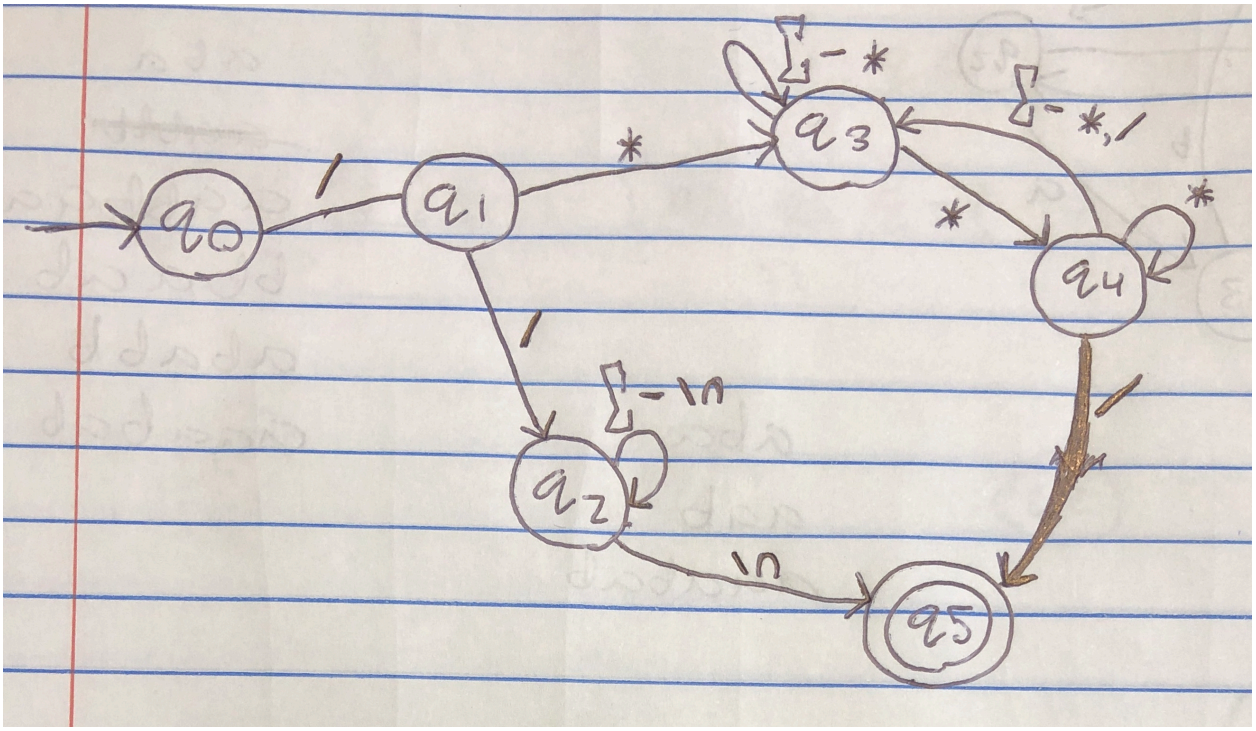
Regular Expression**: ((b + aa)b + (ab + (b + aa)a)(bb + baa)*(a + bab))***

G) {w ∈ {a, b} ∗ | #a(w) − 2#b(w) = 0}. This is not regular because it is not in the language.



H) {w ∈ Σ ∗ | w is a C comment}

The following is the minimal DFSM for the given language. Note, there are no redundant or lost states.

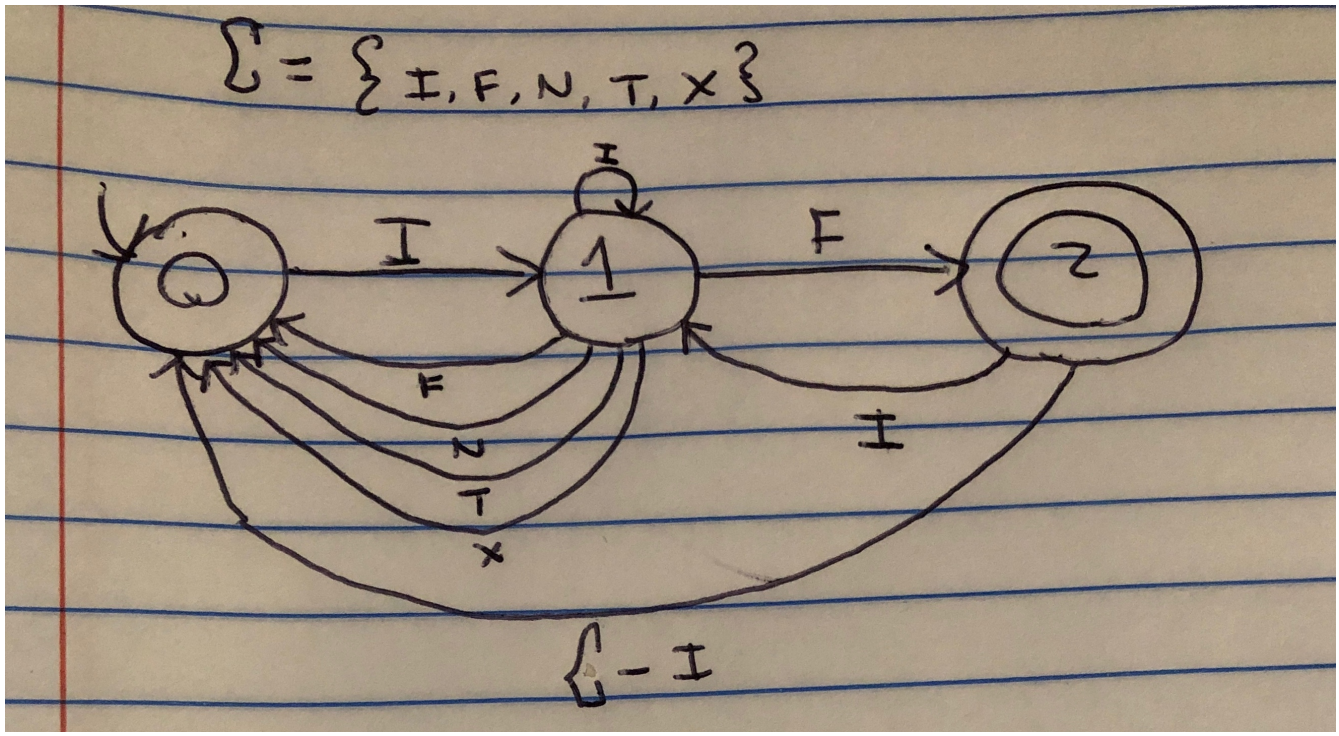


<u>NOTE for the purpose of this regular expression I am making the Kleene star = * and the asterisk in the comment as the $ symbol TO AVOID CONFUSION!!</u>

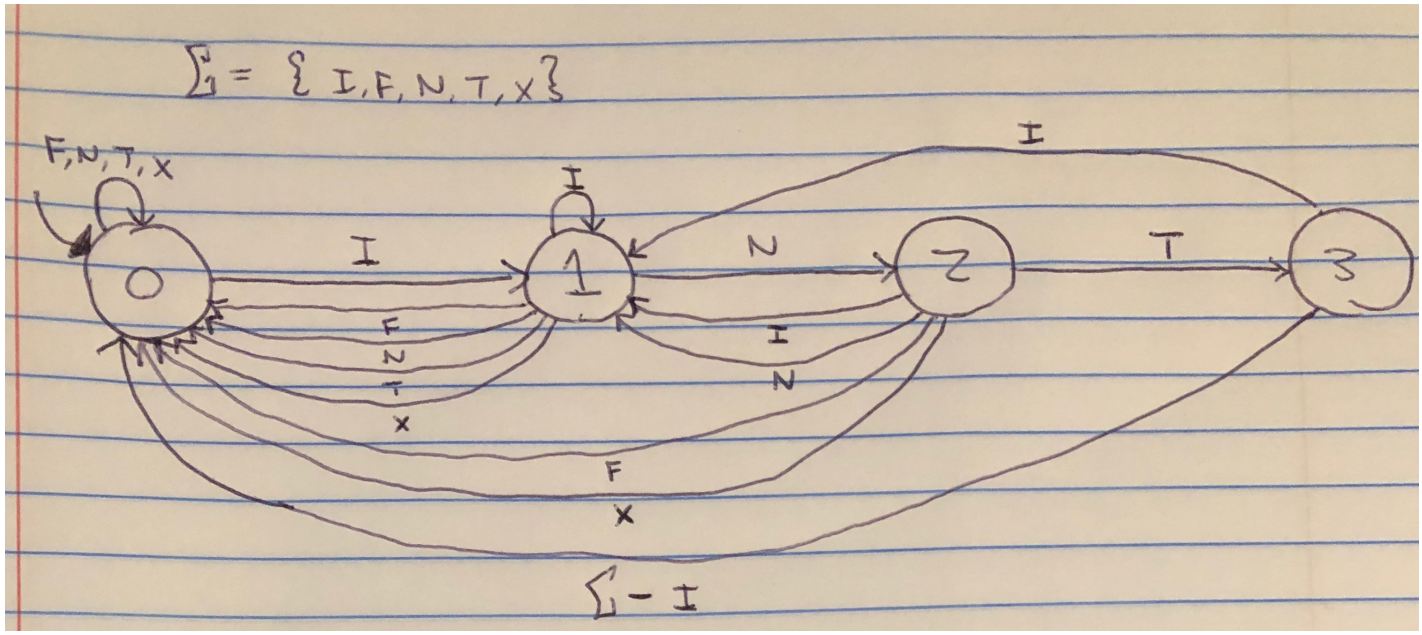
Regular Expression: **(/$(Σ′ − $)*$($ + (Σ′ − {/, \n})(Σ′ − $)* $)* /) + (//(Σ′ − \n)* \n)**

2. The muli-pattern searching problem for the text $T \in \Sigma *$ where the set in the language is {I, f, n, t, x} can be constructed using the minimal DFSM (the NDFSM results in each of the constructions for p1 and p2 directly).

For p1 = if, the following machine can be used to traverse for the sought-after string "if" at linear time complexity:



For p2 = int, the following machine is given:

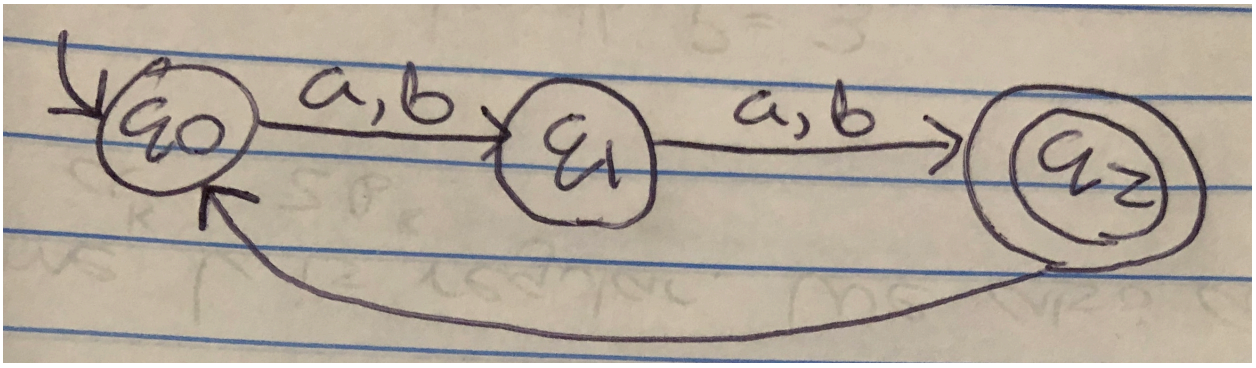


3. Given Σ = {a, b} and α a regular expression, is it true that L(α) contains only non-empty even-length strings in Σ∗ and no string consisting only of b's?

To decide whether the language given by L(α) is decidable or not, we must solve the question that is whether, given α as input, the language *L* accepts the input or not. Therefore, one way of showing that the language L(α) is decidable is by describing an FSM that accepts it.

Therefore, by using *ndfsmsimulate*, we can create an FSM and then use *decideFSM(M, w)*. If *ndfsmsimulate* returns accepts the string given by L, then return True. Otherwise, return false.

Here is a given NDFSM which will calculate even-length strings:



However, as can be seen, this language contains **all** strings of even-length and therefore *must* also contain some strings of only b's. Therefore, no language with these properties exist. As a result of using *decideFSM(M, w)*, it will return false. Therefore, the language *is* decidable.