Solutions of Assignment #1 --- CSC320, Summer, 2018

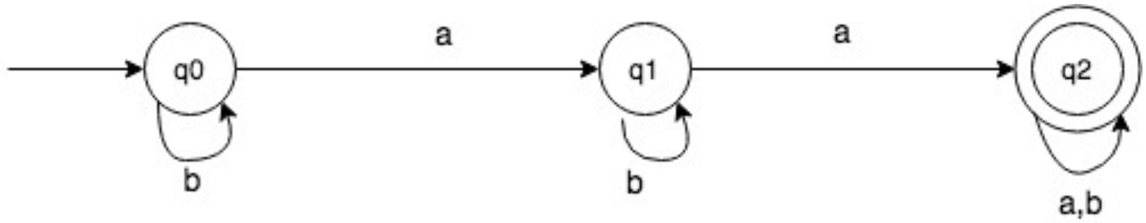
Zhaocheng Li

V00832770

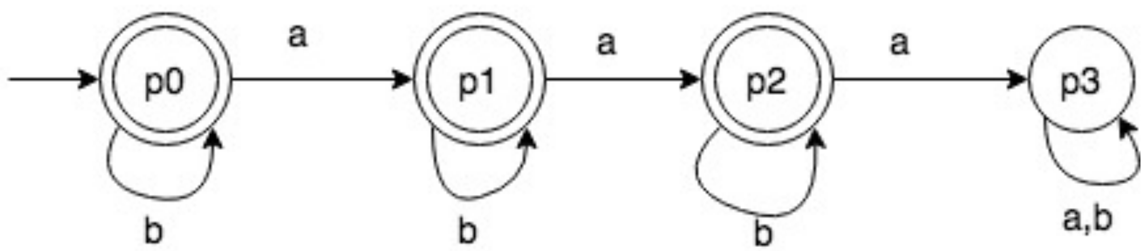
Q#1:

Given the languages L1, L2 over {a, b}*, where L1 contains at least 2 a's and L2 contains at most 2 a's. In this case,

a. We know from L2 that the DFA N1 accepts L1 if and only if it has two or more a's. Hence, we have



b. Similarly, L2 allows at most 2 a's, which means we need to take some extreme cases such as empty string or string with all b's in to consideration. Hence, we have

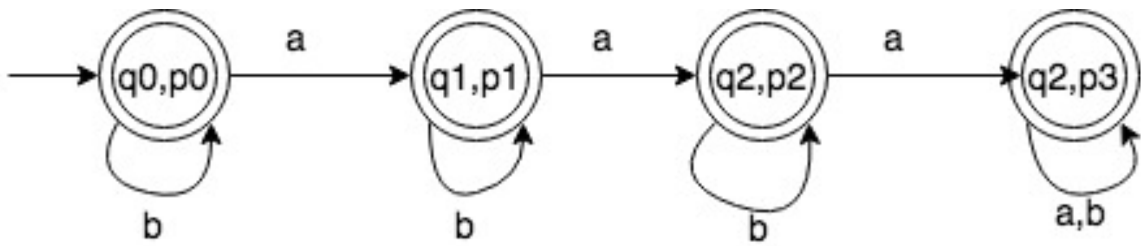


c. Here we are combining N1 and N2 to build a DFA N such that it accepts L1uL2 whenever N1 or N2 accept the input. Hence, we have the set of states (accessible and inaccessible) Q defined as follow:

Q = {{q0, p0}, {q0, p1}, {q0, p2}, {q0, p3}, {q1, p0}, {q1, p1},

{q1, p2}, {q1, p3}, {q2, p0}, {q2, p1}, {q2, p2}, {q2, p3}}.

And such DFA N that accepts L1uL2 is:



Q#2:

Firstly this is the E-table:

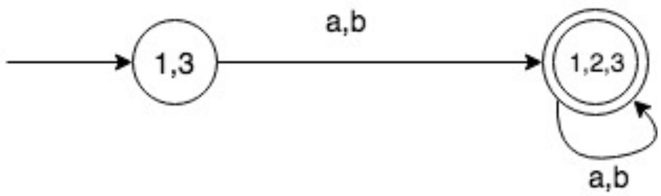
E({1})	{1,3}
E({2})	{1,2,3}
E({3})	{3}
E({1,2})	{1,2,3}
E({1,3})	{1,3}
E({2,3})	{1,2,3}
E({1,2,3})	{1,2,3}

The start state is E({1}) = {1,3}, and we have the transition table:

	a	b

{1,3}	{1,2,3}	{1,2,3}
{1,2,3}	{1,2,3}	{1,2,3}

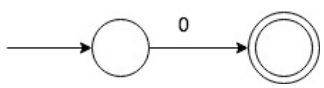
Then we have the diagram:



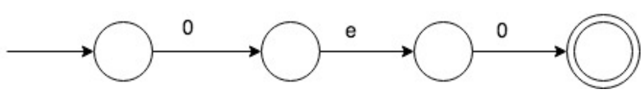
Q#3:

By the technique learnt in class, we show It step by step: (“e” means “ε” here due to limited type of special characters)

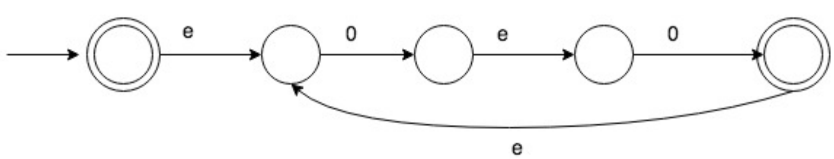
1. “0”



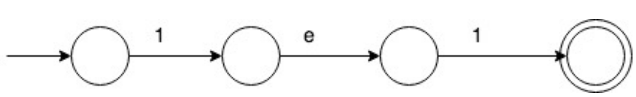
2. “00”



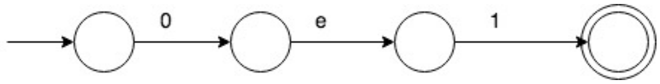
3. “(00)*”



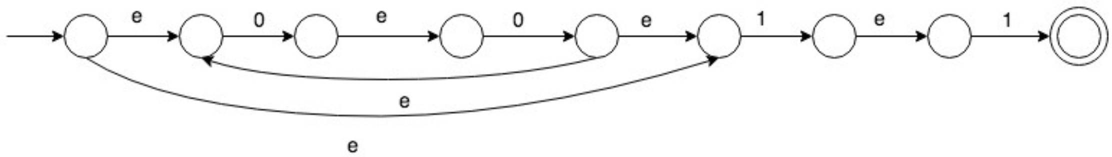
4. “11”



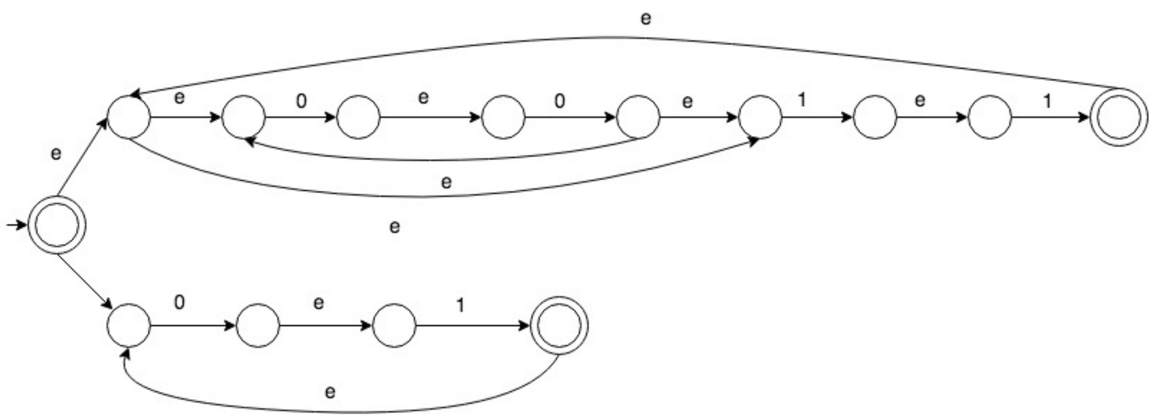
5. “01”



6. "(00)*(11)"



7. "(((00)*(11))U01)*"

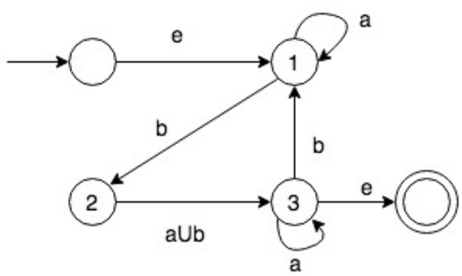


And the final step is the NFA for the given regular expression.

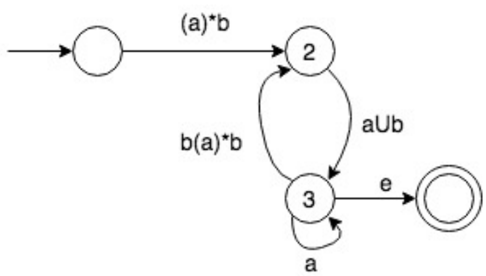
Q#4:

following the "routine": First of all, we need to convert DFA into GNFA, and tip off a state one at a time until there is only start state and accept state left in GNFA, while adjusting the graph all the time. ("e" means "ε" here due to limited type of special characters)

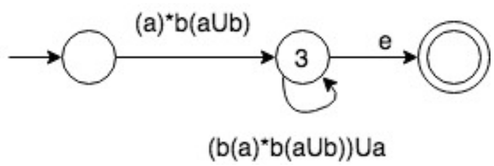
1. we have GNFA:



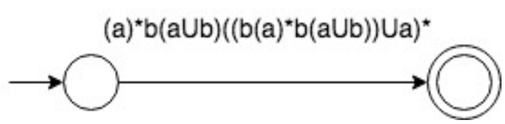
2. Then we rip off state 1 :



3. Rip off state 2:



4. Rip off state 3:



Then the regular expression is shown on the last graph, which is:

*(a)*b(aUb)((b(a)*b(aUb))Ua)**

Q#5:

For language A and B, define the interleave of A and B to be the language:

$\{w \mid w=a_1b_1\ldots a_kb_k$ where $a_1\ldots a_k \in A$ and $b_1\ldots b_k \in B$, and $a_i, b_i \in \Sigma, k \geq l \geq 1\}$

Then we prove by construction NFA N that recognizes the interleave of A and B.

Suppose N={Q, Σ, δ, q, F}, and notice that N need to keep track of the input since we need to always switch between A and B after each character of input is read. Assuming there are DFAs Da for A and Db for B, then the input should be accept by N in the case that bath Da and Db also reach the accept state.

Also remind that N should also accept the empty string.

Formally, we define NFA N as follow, given Da=(Qa, Σ, δa, qa, Fa) and Db=(Qb, Σ, δb, qb, Fb):

1. Q = (Qa x Qb)U(q0), where q0 is the state when nothing is read, and the cartesian product Qa x Qb are states keeping track of both states in Da and in Db.
2. q = q0
3. δ:
   - δ(q0, ε)=(qa,qb),
   - (δa(q,a),p) $\in$ δ((q,p),a), where q is the current state of Qa and p is the current state of Qb,
   - (q, δb(p,a)) $\in$ δ((q,p),a)
4. F = (Fa x Fb)U{q0}, where F accepts when both Da and Db

reach the accept state, or the empty string is accepted.

And by the formal definition of NFA N for the interleave of A and B, the regular languages are proved to be closed under the mix operation.