Jonathan Tso
HW02
CS 301

1. Final

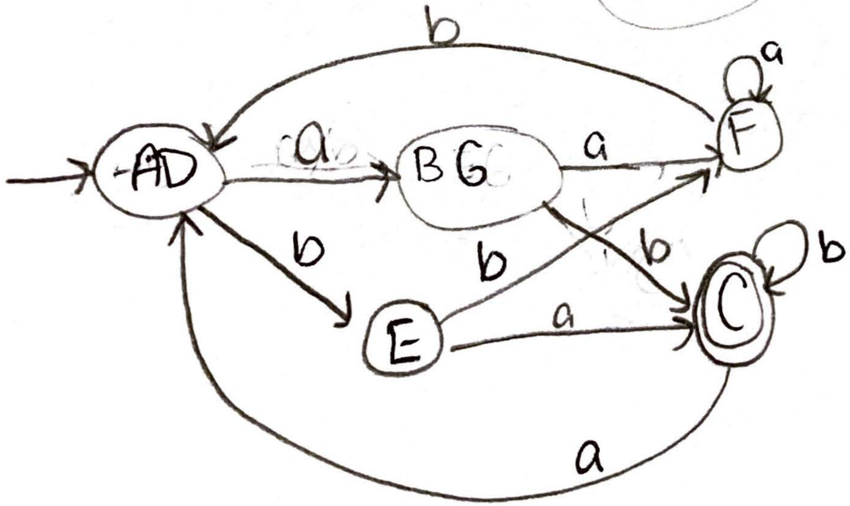
C

Non Final

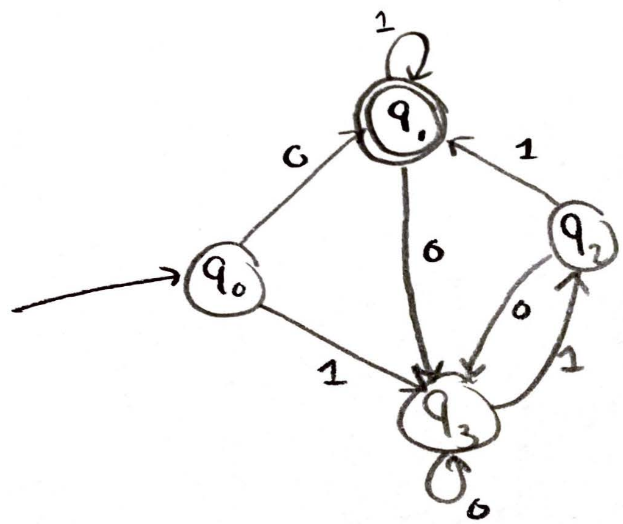
AB DEFG

AD F

BG E states that lead to final E diff



2.



Setting an arbitrary string S to be the possible path(s) from q_3 to q_1

$$S = (0)^*(10)^* \,||\, (1)^*$$

we can go from q_0 to q_1 with the regex expression:

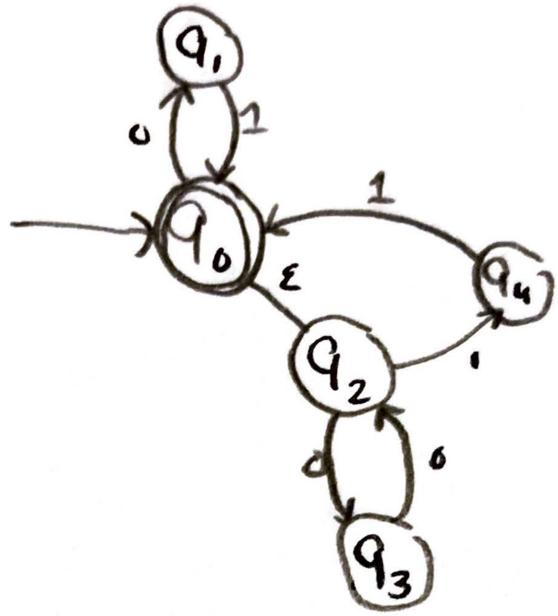
$$\emptyset(1)^*(\emptyset S)^* \,|\, 1S(\emptyset S)^*$$

3. For regexes $(AB \mid A)^* A$ and $A(BA \mid A)^*$, we know that due to the Kleene star, we can run ∅ to infinite iterations of what in parenthesis. Assuming ∅ iterations, we can see that for both regex, the minimum to be accepted is string = A. We can also see that on the left side, it must end in A, and B must always have a preceding A. This can be formally true on the right hand regex as well. On the right hand side, the language must always start with A, similar to the left hand side. This means if we pull BA first, it will have a preceding A. Additionally, every time B is called, it follows with an A, so any extra B will also have a preceding A. Another result of this is that it will always end with A, similar to the left regex. Finally, both regex give the option to run A an infinite number of times due to the Kleene star and being in the or parenthesis. Thus, all languages formed on the left and right regex are equal.

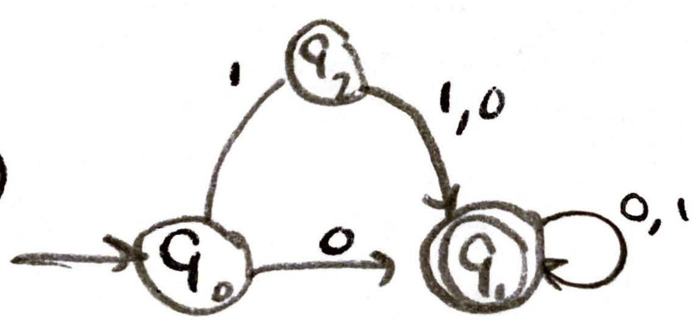
b. $(A \mid B)^* B = (A^* B)^* B$

Consider the string = AB. On the left hand regex, this is an accepted string of the language. However, the right hand regex does not accept because for any instance of A, we must follow with a B and it ends with an additional B. Thus, these two regex are not equal.

4. a)



b)



5. Suppose that extended regex produce regular languages. This, then, must suppose that intersection and complement, which are included, are closed and themselves produce regular languages. We will prove both individually, thus showing that all extended regex produce regular languages as a result of using all closed operations.

Intersection

Suppose we have DFA L_1 and L_2, that recognize a language and are thus regular. As provided in class in lecture Sept 10, we can construct a unity DFA that encompasses L_1 and L_2 by running them in tandem. The new DFA will record the states of both DFA. An example DFA is provided under the "Closure" section of the Sept 10 lecture. From this, we can see that there is a corresponding accept state(s) for every instance both L_1 and L_2 are both in an accept state. This DFA generated can decide a language of the intersection of L_1 and L_2 => because a DFA can decide the language of two regular languages, we can say the intersection is closed and thus produces regular languages.

Complement

Suppose we have an NFA such that it has an initial state, normal states, and final state(s). We can generate a complement by turning all non-final states into final, and final states to non-final. What this approach does is provide an accept for every instance that was not part of the accepted language before, and no longer accepts any string that previously was accepted, thus providing the complement of accepting the original language. This is also in part due to the fact that we know each existing state has a transition initially, so all states and transitions still exist there. Thus, we can produce regular languages out of the complement.

Thus, by proof of intersection and complement being closed and produce regular languages, and that regex are equally expressive as DFA/NFA, extended regex produce regular languages.

5. b) Prove if an GNFA exists and decides language L, then L is regular.

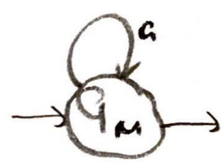
Suppose that the GNFA that decides language L exists. We can determine that the GNFA has an accept state, start state, and transition from q_0 to q_F via regex. For each character in the regex, we can make a transition interim between q_0 and q_F in order such that the character of the regex is the transition.

GNFA $\quad q_0 \xrightarrow{(\Sigma)^n} q_F$

where n = the number of characters in the language that may be accepted. In other words, the full regex can be represented as a sequence of Σ up through satisfaction.

By breaking apart the regex into all individual characters inside the alphabet of the language Σ, we can generate an equal number of n states. This is supplanting the original transition from q_0 to q_F with directionally relevant transitions that will still fulfill reaching q_F. We know that can exist as individual transitions since we have already discussed the relevant method of converting an NFA to GNFA is trivializing each transition until we reach the full string. Here, we are applying the inverse until all our transitions are the empty string or alphabet single character. As a result, we can generate an expressive NFA for such a GNFA. As previously discussed, if a language can be decided by an NFA, we can determine the language is regular. Thus, by proving we can generate an NFA from the GNFA, we have proven the decided language is regular.

6. Suppose we have a language that is decided by an NFA. Suppose that the corresponding # of regex for the NFA is finite. This means that the number of regex can be counted. This can be disproven when considering any NFA that is inclusive of an * or Kleene star. Consider a state q_M that is in the middle of an NFA, which loops into itself on a of Σ.



This can be written as the following... a^* or a, or aa, aaa, ...

Thus, we can see that while the Kleene provides a succinct representation, we have an infinite number of regex since the Kleene can be applied an infinite number of times.