COT 4210 Quiz #1A Solutions

2/4/2021

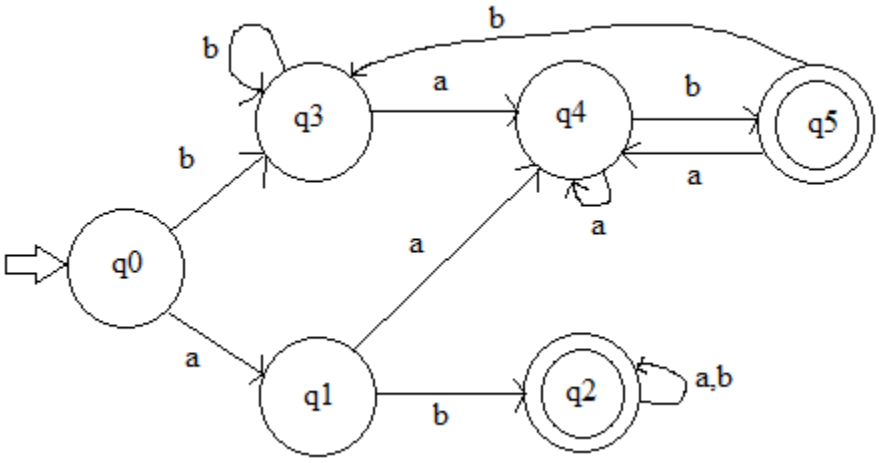
1) (9 pts) Design a DFA over the alphabet Σ = {a, b} that accepts all strings that either start with ab or end with ab. Please clearly mark the start state, all accept states, and all transitions with clearly labeled arrows. (Examples of strings in the language are abbbbbbabba, bbaaab and abab. Examples of strings not in the language are a, b, and aabba.)

Solution

The key to this design is creating a pathway for all strings that start with ab. Then, knowing that the machine has to be a DFA, add the appropriate transitions to make the machine a valid DFA, which will create unique paths for all strings that do NOT start with ab. In addition, we must keep track of how far along the progression of creating ab we are. So, if the last character is an a and we didn't start with ab, we have a non-accept state that designates that we just need an b to be accepted. We have another state that designates that we didn't start with ab and we need both letters to end our string with ab (so our previous letter was a b). So, our states are:

q0 = start state, nothing has been read in, we never go back to this state.
q1 = we have read in a single a and nothing else, we never go back to this state.
q2 = our string started with ab, we can just stay in this state since the string must be accepted no matter what comes later. (Accept state)
q3 = our last character read in is a b and we haven't started or ended in ab.
q4 = our last character read in was a and we didn't start with ab
q5 = our last two characters read in were ab. (Accept state)

Here is the picture:

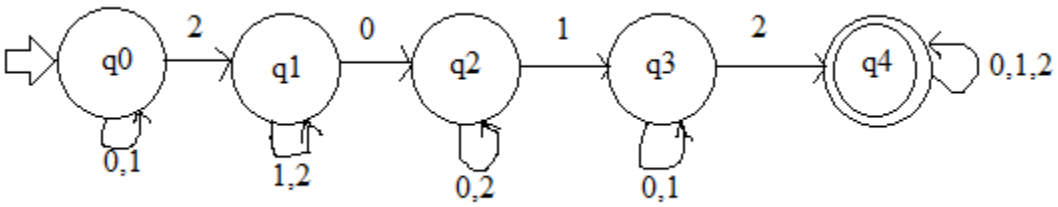


Grading Criteria:
1 pt clearly marked start state
1 pt clearly marked accept state(s)
2 pts all valid transitions are shown (give 1 pt if most are there)
2 pts for accepting all strings that start with ab
2 pts for accepting all strings that end with ab
1 pt for rejecting all strings that should be rejected

2) (9 pts) Design an NFA over the alphabet Σ = {0, 1, 2} that accepts all strings that contain the subsequence 2, 0, 1, 2. Note: a subsequence is a subset of items in a sequence which appears in the same order within the larger sequence. For example, the sequence 0, 0, 0, 1, **2**, 1, **0**, 2, **1**, 0, **2**, 1, contains the subsequence 2, 0, 1, 2 (highlighted). Please clearly mark the start state, all accept states, and all transitions with clearly labeled arrows. (An example of a string not in the language is **2**22221111111**0**0000022222**1**111100001. The bold characters identify the "best choice" for the subsequence, but there is no 2 following the bolded 1, which is why the string isn't in the language.)

Solution

Since this is an NFA, just create a straight path left to right with different states upon reading in 2, 0, 1 and 2, respectively and keep Σ loops everywhere else. Also, you can make this a DFA by realizing that you stay at a state until you "get" the correct character. Here is one possible solution:



Grading Criteria: **1 pt clearly marked start state**
1 pt clearly marked accept state(s)
4 pts for accepting any string that should be accepted
3 pts for rejecting all strings that should be rejected

3) (6 pts) Give a regular expression for the language of all strings over the alphabet {a, b} that does NOT contain two consecutive a's. (Examples of strings that belong to this language are abbb, ababa, and bbbb. Examples of strings not in the language are: bbbbaa, abaa, and abbbabaabb.)

Solution

The string can start and end with an a optionally, followed by 1 or more b's followed by at most 1 a, repeated as many times as desired. Here is a valid regular expression that encapsulates this idea:

$$(\boldsymbol{\varepsilon} \cup \boldsymbol{a})(\boldsymbol{bb}^*(\boldsymbol{\varepsilon} \cup \boldsymbol{a}))^*$$

So, I allow a string to end or not end in a by grouping a with epsilon in an union after completing 1 or more b's.

Grading criteria: **2 pts for using only valid R. E. rules (so no +, no powers to an integer)**
2 pts for rejecting all strings that have the substring aa
2 pts for accepting all other strings