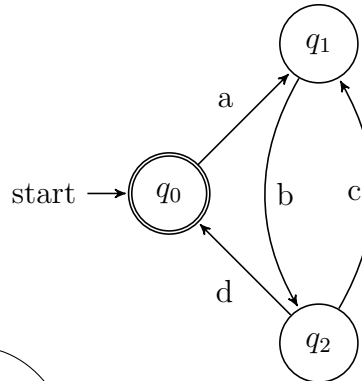


Question 1. Give a regular expression for the language accepted by the following NFA N :



Thoughts:

Best method is probably to look at strings that are in $L(N)$ and hypothesize the answer.

We are not expected to memorize long algorithms like nfa-to-reg conversion.

Answer:

We explore what strings are accepted by N to deduce the language $L(N)$.

$\lambda \in L(N)$ - There is an obvious kleene star cycle, that begins with a and ends with d .

$abd \in L(N)$ - There must be a b after the first a and if there is a c , it must be followed

$abcbd \in L(N)$ by b .

$abcbcbcbd \in L(N)$

$abcbcbdabcbd \in L(N)$

The regular expression $r_N = (ab(cb)^*d)^*$ satisfies $L(r_N) = L(N)$. An alternate answer can be $r_N = (a(bc)^*bd)^*$.

Question 2. Give a regular expression for the language $L = \{w \in \Sigma^* \mid |w| \text{ is even OR } w \text{ starts with an } a\}$.

Thoughts:

One way to start is to address “ $|w|$ is even” and “ w starts with an a ” separately. In this case the **OR** makes a merger of the two sub regular expressions easy.

Answer:

Part 1:

$|w|$ is even : $(aa + bb + ba + bb)^* = ((a + b)(a + b))^*$

Part 2:

w starts with a : $a(a + b)^*$

So $r_L = ((a + b)(a + b))^* + a(a + b)^*$ is a regular expression for the language L because **OR** can be represented by $+$.

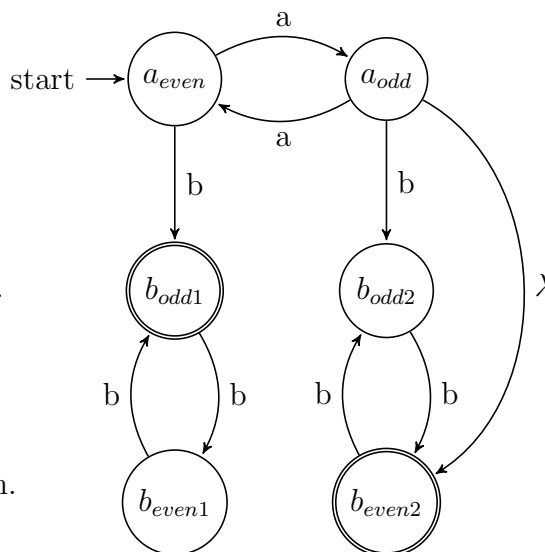
Alternatively $r_L = (aa + ab + ba + bb)^* + a(a + b)^*$ is also a regular expression for L .

Question 3. Give a NFA accepting the regular language $L = \{a^i b^j \mid i, j \geq 0 \text{ and } |i - j| \text{ is odd}\}$.

Thoughts:

" $|i - j| \text{ is odd}$ " feels similar to " $|w| \equiv k \pmod t$ " type of languages. In fact " $|i - j| \text{ is odd}$ " is the same as $|i - j| \equiv 1 \pmod 2$. We can try a similar approach. A good idea with every question is to see if there are similarities or patterns relative to other problems you have solved.

NFA N :



Answer:

To keep track of the difference between the number of a 's and b 's being odd ($|\#a - \#b| \equiv 1 \pmod 2$) we first keep track of the parity of the a 's and then accept on the opposite parity for the b 's.

- | even a 's – even b 's | is even.
- | odd a 's – odd b 's | is even.
- | even a 's – odd b 's | is odd.✓
- | odd a 's – even b 's | is odd.✓

We have to be able to accept on an odd number of a 's and zero b 's so there is an extra lambda transition added. The NFA $N = \{Q, \Sigma, \delta, a_{\text{even}}, F\}$ where $\Sigma = \{a, b\}$ described by the diagram above accepts the language L .

Question 4. Give a **succinct** description of the language $L = L(G)$ generated by the grammar G containing the following seven productions:

$$S \rightarrow aSb \mid A \mid B; \quad A \rightarrow aA \mid a; \quad B \rightarrow bB \mid b.$$

Thoughts:

Looks like $a^i b^j$ with some condition at first glance. I would try example strings to figure out the conditions.

Answer:

We explore what strings belong to the grammar G so that we can figure out the language $L(G)$.

$\lambda \notin L(G)$ Only strings that are composed of strictly more a 's or more b 's are produced
 $a \in L(G)$ by the grammar. a 's and b 's can be produced together from the center of the
 $b \in L(G)$ string and then either more a 's or more b 's.

$aab \in L(G)$

$abb \in L(G)$ The language $L(G)$ that is described by G is :

$$L(G) = \{a^i b^j \mid i, j \geq 0, i \neq j\}.$$

$ab \notin L(G)$

$aabb \notin L(G)$

$aaabbb \notin L(G)$

Question 5. Give a minimal DFA that has exactly two final states.

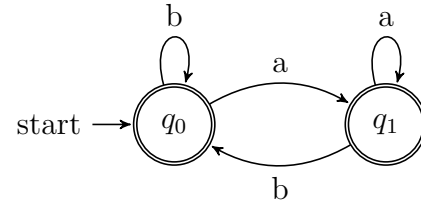
Thoughts:

Probably looking for knowledge of:

- minimal
- DFA and
- final states .

Natural First Try

M :

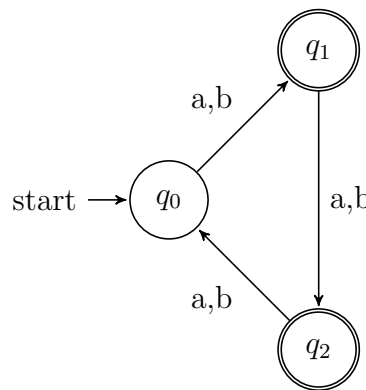


M is a legal DFA over the alphabet $\Sigma = \{a, b\}$, there are no λ transitions, and every state has a path for all possible inputs. Now is minimality addressed? Well this DFA accepts all strings over Σ . The minimal DFA for the language $L((a + b)^*)$ has one state and thus M has one indistinguishable state and is not minimal. Now you might think what if we made the alphabet $\Sigma = \{a, b, c\}$? To keep M a legal DFA, c would have to be accounted for as input when the automaton is in state q_0 or q_1 . Again we create a DFA that accepts all strings since every state is an accepting state, and therefor not minimal.

Answer:

The next natural try is a DFA with three states, keeping it simple we will define it over $\Sigma = \{a, b\}$. There are several that work, but to avoid the risk of the DFA not being minimal, we can create a DFA that considers strings whose lengths are multiples of three. Since we need two accepting states, we can create a DFA for the language $L = \{w \in \Sigma \mid |w| \not\equiv 0 \pmod 3\}$.

D :



There are no indistinguishable states, each state keeps track of the remainder of the length of the string when divided by three. Thus D is a proper minimal DFA, and has two final states.

Question 6. Give an infinite regular language L_{reg} such that

$$L_{\text{reg}} \subseteq \{w \in \{a, b\}^* \mid |w|_a = |w|_b\} = L.$$

Thoughts:

- L has all strings with the same number of a's and b's
- L is non regular
- Kleene star is your key tool for finding infinite regular languages
- Explore by writing down different infinite subsets of L .

Subsets of L include:

$$\begin{aligned} &\{a^i b^i \mid i \geq 0\} \\ &\{a^i b^i \mid i \equiv 0 \pmod{2}\} \end{aligned}$$

All are non regular still, but with the use of Kleene star its actually easy to find an infinite regular subset. Just follow the rule that for every b introduced, there has to be an a . This guarantees the same number of a 's and b 's.

Answer:

The set $L_{\text{reg}} = L((ab)^*)$ is an infinite regular subset of L . Alternatively all of

$$L_{\text{reg}} = L((abab)^*), L_{\text{reg}} = L((aabb)^*), L_{\text{reg}} = L((babbabbaaa)^*)$$

and so on will work.

Question 7. (Thinker) Give a non-regular language L such that $L \subseteq L(b^*)$.

Thoughts:

Since there is only one symbol think about what patterns DFA's capture. DFA's have a graph structure and paths. Cycles in DFA's allow for infinitely **many** different strings to be accepted. We can exploit that cycles are limited to capturing a certain type of pattern - namely a linear equation type pattern $ax + b$.

Answer:

Let $L_p = \{b^i \mid i \text{ is prime}\}$ is a non-regular subset of $L(b^*)$. The difference between any two strings in L_p is not linear, but grows at about $\frac{x}{\log(x)}$ as x goes to ∞ by the prime number theorem. Other examples include:

$$L = \{b^i \mid i \text{ is a perfect square}\}, L = \{b^i \mid i \text{ is an element of the Fibonacci sequence}\}.$$