York University
CSE 2001 – Unit 3.1 DFA Classes
Converting between DFA, NFA, Regular Expressions, and Extended Regular Expressions

Instructor: Jeff Edmonds

Don't cheat by looking at these answers prematurely.

1. For each of the following theorems, give a two or three sentence sketch of how the proof goes or why it is not true.

   (a) Every DFA M can be converted into an equivalent NFA M' for which $L(M) = L(M')$.

   - Answer: True. A DFA M is automatically also a NFA.

   (b) Every NFA M can be converted into an equivalent DFA M' for which $L(M) = L(M')$.

   - Answer: True. M' computing is like having many clones computing on M. The current state of M' is the subset of the states of M that the clones are currently on.

   (c) Every DFA M can be converted into an equivalent TM M' for which $L(M) = L(M')$.

   - Answer: True. A TM M' can simulate a DFA M simply by not using its tape except for reading the input in.

   (d) Every TM M can be converted into an equivalent DFA M' for which $L(M) = L(M')$.

   - Answer: False. A TM can compute $L = \{0^n 1^n \mid n \geq 0\}$ and a DFA cannot.

   (e) Every regular expression R can be converted into an equivalent NFA M for which $L(R) = L(M)$.

   - Answer: True. NFA are closed under union, concatenation, and kleene star. Hence, the NFA M is built up by following these operations within the R.

   (f) Every DFA M can be converted into an equivalent regular expression R for which $L(M) = L(R)$.

   - Answer: True. This is the complex algorithm in which states of the NFA are removed one at a time and edges are allowed to be labeled with regular expressions.

   (g) Every NFA M can be converted into an equivalent one M' that has a single accept state.

   - Answer: True. Given M, construct M' by adding a new state f that will be the only accept state of M'. Add an ϵ transition from each accept state of M to f.

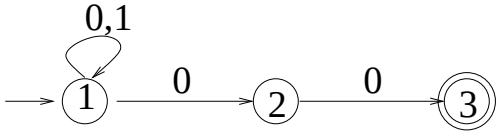
   (h) The set of languages computed by DFA is closed under complement.

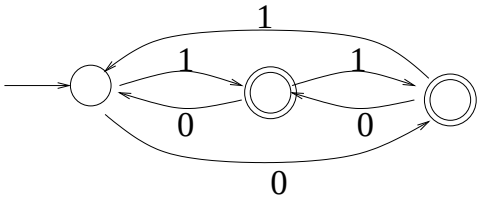
   - Answer: True. Given a DFA M that computes L, construct M' by switching the accept and not accept states of M. $L(M') = \overline{L(M)}$.

   (i) The set of languages computed by NFA is closed under complement.

   - Answer: True. Given a NFA M that computes L, constructing M' for which $L(M') = \overline{L(M)}$ cant be done directly. Instead, convert the NFA M into a DFA M'' as done above and then construct from the DFA M'' the DFA M' by switching the accept state, so that $L(M') = \overline{L(M'')} = \overline{L(M)}$.

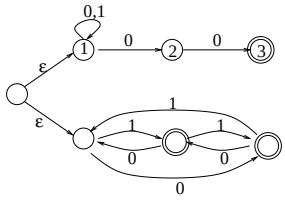
2. Closure: Consider the automata from the last assignment.





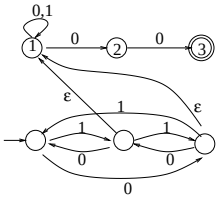
(a) Construct an NFA for the language $L_1 \cup L_2$. Use the technique done in class that combines the above two machines. Do not simplify the machine produced.

- Answer:



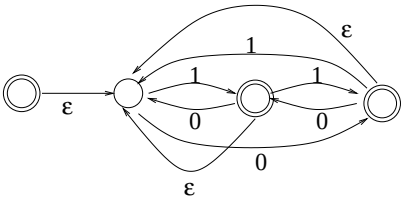
(b) Similarly, construct an NFA for the language $L_2 \circ L_1$.

- Answer:



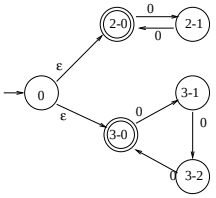
(c) Similarly, construct an NFA for the language $(L_2)^*$.

- Answer:



(d) Suppose you are a DFA. You have an NFA M that accepts L and an input string ω. In your own words, what are the ideas behind simulating M on ω? What states to you need?

3. Consider the NFA M:

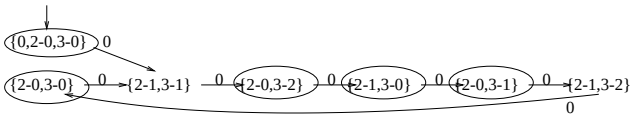


(a) Give the language $L(M)$.

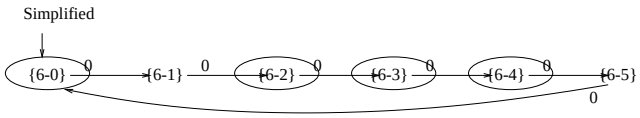
- Answer: $L(m) = \{w \in \{0\}^* \mid |w| = 0 \ mod \ 2 \ or \ |w| = 0 \ mod \ 3\}$

(b) Convert this NFA into a DFA. Giving the table, the DFA M', and the simplified DFA.

- Answer:



$Q\backslash\Sigma$	0
0	$\emptyset$
$2-0$	$\{2-1\}$
$2-1$	$\{2-0\}$
$3-0$	$\{3-1\}$
$3-1$	$\{3-2\}$
$3-2$	$\{3-0\}$



(c) Make a DFA M'' for the following language $L'' = \{w \in \{0\}^* \mid |w| \text{ is 0, 2, 3, or 4 mod 6}\}$.

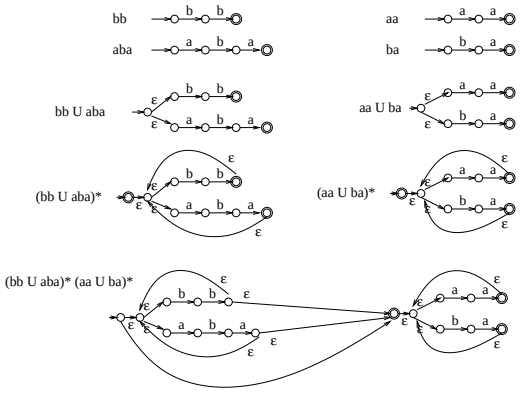
- Answer: See above fig.

(d) Is there a connection between M' and M''? Why? (If you are in the mood, see references for the Chinese Remainder Theorem of number theory.)

- Answer: They compute the same language. The Chinese Remainder Theorem tells you for example that if $x = 1 \; mod \; 2$ and $x = 2 \; mod \; 3$ then $x = 5 \; mod \; 6$.

4. Construct an NFA for the following language $(bb \cup aba)^*(aa \cup ba)^*$.

- Answer:



5. Use the Bumping Lemma to prove that the following is not regular.
$L = \{a^n b a^m b a^{n+m} \mid n, m \geq 0\}$

- Answer: This L is not regular because it distinguishes between the infinite number of prefixes in the set $S = a^*$. The prefixes $\alpha = a^i$ and $\beta = a^j$ are distinguished by L as follows. Let $\underset{\sim}{=} baba^{i+1}$, then $\alpha \underset{\sim}{=} a^i baba^{i+1}$ is in L and $\beta \underset{\sim}{=} a^j baba^{i+1}$ is not.

6. Algorithms: Describe in a few sentences the outline of an algorithm to solve each of the following computational problems involving DFA and NFA.

(a) Given an NFA M, does it accept any string or is it the case that $L(M) = \emptyset$.

- Answer: An NFA M accepts the string α iff there exists a path from the start state to an accept state labeled α. There is some string that M accepts iff there exists a path from the start state to an accept state. To check this, we consider each accept state one at a time and ask whether there exists a path from the start state to this accept state.
When the NFA M is viewed as a directed graph, this question can be rephrased as the following classic graph theory problem. The input consists of a directed graph and two nodes s and t. The output states whether or not there is a path in the graph from s to t. There are standard algorithms for this problem.

(b) The symmetric difference of two languages is defined to be $L_1 \oplus L_2 = (L_1 \cap \overline{L_2}) \cup (L_2 \cap \overline{L_1})$. It consists of those strings for which these languages give different answers.

Given two DFA $M_1 = \langle Q_1, \Sigma, \delta_1, s_1, F_1 \rangle$ and $M_2 = \langle Q_2, \Sigma, \delta_2, s_2, F_2 \rangle$, construct a DFA $M = \langle Q, \Sigma, \delta, s, F \rangle$. Then formally prove as done in class that $L(M) = L(M_1) \oplus L(M_2)$, i.e. that for every string α, $\alpha \in L(M)$ if and only if $\alpha \in L(M_1) \oplus L(M_2)$.

- Answer: Define $M = \langle Q_1 \times Q_2, \Sigma, \delta, \langle s_1, s_2 \rangle, F \rangle$, where $\delta(\langle q_1, q_2 \rangle, a) = \langle \delta_1(q_1, a), \delta_2(q_2, a) \rangle$ and $F = \{\langle q_1, q_2 \rangle \mid (q_1 \in F_1 \text{ and } q_2 \notin F_2) \text{ or } (q_1 \notin F_1 \text{ and } q_2 \in F_2)\}$. We prove $L(M) = L(M_1) \oplus L(M_2)$ as follows. $w \in L(M)$ iff M computing on w halt in some state $\langle q_1, q_2 \rangle \in F$ iff M_1 and M_2 computing on w halts in states such that one is accepting and the other is not iff $w \in L(M_1) \oplus L(M_2)$.

(c) Given two NFA M_1 and M_2, determine whether $L(M_1) = L(M_2)$.

- Answer: Observe that $L_1 = L_2$ if and only if $L_1 \oplus L_2 = \emptyset$. We test this as follows. First covert each NFA into equivalent DFAs using 2^Q clone method. Then uses algorithm from (b) to construct a DFA M for the symmetric difference of their languages. Finally, use algorithm from (a) to test whether M accepts any strings.

(d) Given an NFA M, determine whether $L(M) = \{\omega \mid \omega \text{ contains 0101 as a substring }\}$.

- Answer: Let M_2 be the NFA given in the notes accepting the language $\{\omega \mid \omega$ contains 0101 as a substring $\}$. Use algorithm from (c) to test whether $L(M) = L(M_2)$.

(e) Given an NFA M, determine whether $L(M) = \{0^n 1^n \mid n \geq 0\}$.

- Answer: No NFA accepts this language. Hence, when we are given an NFA, we can simply say NO without even looking at it.

7. Let L be a language of strings from $\{0,1\}^*$.
We say that the strings α and β are distinguished by L if there exists a γ such that $L(\alpha\gamma) \neq L(\beta\gamma)$. Don't try to prove it, but what did we say in class that this says about any DFA computing L?

Give a first order logic statement that states that the strings α and β are **not** distinguished by L. Don't try to prove it, but what did we say in class that this says about any DFA computing L?

Our L happens distinguish between every pair of strings in the set $S = \{0100, 1001, 0010\}$. On the other hand, L does not distinguish between the strings $\epsilon, 0100, 01000, 10011, 00100$. Neither does it distinguish between $1001, 01001$. Neither does it distinguish between $0010, 10010, 00101$. The string 11111 happens to be accepted in L, but strings 00000 and 10010 are rejected.

Surprisingly enough, this is enough information about the language L to completely determine what answer it gives for every binary string. More over, this specified language is regular and has a very simple DFA. With a small change, this is proved in Eric's email to me in my 2001 course notes. You do not need to read or understand this proof unless you want.

All you need to do is to use the above information to figure out what this DFA must look like.

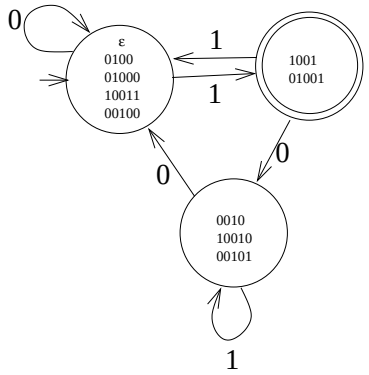
- Answer: We proved in class that if L distinguishes between two strings then these two strings **need** to go to the different states in any DFA that computes L. Because L distinguishes between the three strings in S, they must go to three different states. Lets call these states q_{0100}, q_{1001}, and q_{0010}.

  We say that the strings α and β are **not** are distinguished by L if $\forall\gamma,\ L(\alpha\gamma) = L(\beta\gamma)$. Given a "past" of α or of β, all futures γ lead to the same result. A DFA computing L does not need to have these strings go to the same state, but if they might as well because if they went to different states these two states could be collapsed into the same state. We might as well assume that the strings $\epsilon, 0100, 01000, 10011, 00100$ all go to state q_{0100}. Also $1001, 01001$ go to state q_{1001}. Finally $0010, 10010, 00101$ go to q_{0010}. See the states in the figure below.

  Suppose the DFA is in state q_{0100}. As far as the future is concerned we might as well assume that the prefix read so far is 0100. If the DFA then reads the character 0 then it has effectively read

the prefix 01000. We know that 01000 goes to state q_{0100}. This gives us that $\delta(q_{0100}, 0) = q_{0100}$. Similarly, the other transitions are obtained as follows.

Current state	next character	assumed prefix	next prefix	next state
q_{0100}	0	0100	01000	q_{0100}
q_{0100}	1	0100	01001	q_{1001}
q_{1001}	0	1001	10010	q_{0010}
q_{1001}	1	1001	10011	q_{0100}
q_{0010}	0	0010	00100	q_{0100}
q_{0010}	1	0010	00101	q_{0010}



Note this chart is enough to close the DFA, meaning each of its states has an edge labeled 0 and one labeled 1. Unlike Eric's proof, we did not say S was maximal so there could have been more strings indistinguishable from each of these three and hence more states. But because the DFA is closed, this can't be.

Because ϵ is indistinguishable from 0100, it must go to state q_{0100}. Hence this must be the start state.

We were told that the string 11111 happens to be in L. If we run our DFA on this string, it goes to state q_{1001}. Hence this state must be an accept state. Similarly we are told that 00000 and 10010 are rejected by L. They go to the other two states and hence these states must be reject states.

8. The operation of *shuffle* is important in the theory of concurrent systems. If $x, y \in \Sigma^*$, we write $x \parallel y$ for the set of all strings that can be obtained by shuffling strings x and y together like a deck of cards; for example

$$ab \parallel cd = \{abcd, acbd, acdb, cabd, cadb, cdab\}.$$

The set $x \parallel y$ can be defined by induction:

$$\epsilon \parallel y = \{y\},$$
$$x \parallel \epsilon = \{x\},$$
$$xa \parallel yb = (x \parallel yb)\{a\} \cup (xa \parallel y)\{b\}.$$

The shuffle $L_1 \parallel L_2$ of two languages L_1 and L_2 is the set of all strings obtained by shuffling a string from L_1 with a string from L_2:

$$L_1 \parallel L_2 = \cup_{x \in L_1, y \in L_2} x \parallel y$$

For example,

$$\{ab\} \parallel \{cd, e\} = \{abe, aeb, eab, abcd, acbd, acdb, cabd, cadb, cdab\}.$$

Show that if L_1 and L_2 are regular languages then so is $L_1 \parallel L_2$. Do this by describing a general method of constructing an NFA $M_\parallel$ for $L_1 \parallel L_2$ out of DFA M_1 for L_1 and M_2 for L_2.

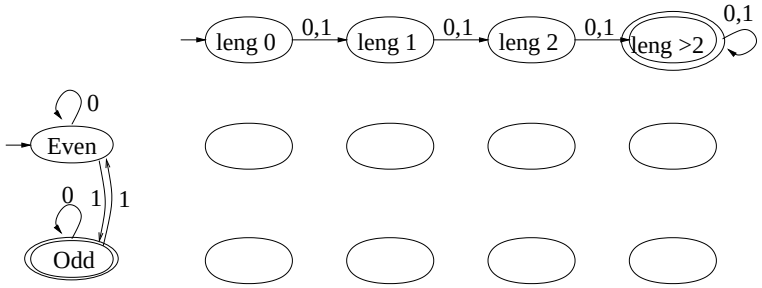
Hint: Given a string γ, we must decide whether or not it is a shuffle of a string α from L_1 and one β from L_2. Given we are building an NFA, we do have a Fairy Godmother to help us. She can tell us for each letter of γ whether it is in α or in β. Then knowing α and β, we can use M_1 and M_2 to see whether or not α is from L_1 and β is from L_2. We accept γ if this the case. On the other hand, we have to run M_1 and M_2 in parallel just as we did when we computed $L_1 \cap L_2$. Towards this goal, imagine putting a pebble on a state of M_1 and another on one of M_2. Guess nondeterministically which pebble to move. Accept if in the end both pebbles occupy accept states.

(a) Now assume M_1 and M_2 are arbitrary DFA. Describe how you would construct the NFA $M_\parallel$.

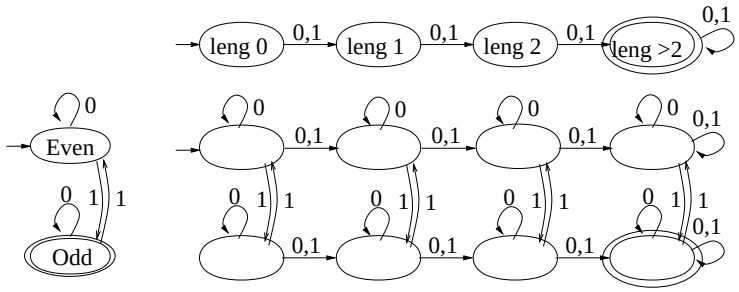
- Answer: For every state u in M_1 and state v in M_2, our new NFA $M_\parallel$ will have a state $\langle u, v\rangle$. The loop invariant and/or interpreted meaning of such a state $\langle u, v\rangle$ is as follows. The NFA $M_\parallel$ has so far read some part of the input string γ. The Fairy Godmother has specified which of these characters are a part of α from L_1 and which are a part of β from L_2. If one would follow this specified α part in the machine M_1, then the resulting state would be u. Similarly, if one would follow this specified β part in the machine M_2, then the resulting state would be v.

  When the next character of γ is read, the Fairy Godmother tells us whether this letter is the next in α or in β. We then either follow the edge labeled with this character in M_1 or in M_2. This is done more formally as follow. Consider any state $\langle u, v\rangle$ of the new NFA $M_\parallel$ and any character $c \in \Sigma$. Let u' denote the state of M_1 reached when traveling from state u and reading this character c, i.e. using transition rule $\delta_1(u, c) = u'$. Similarly let v' denote the state of M_2 reached when traveling from state v and reading c, i.e. using transition rule $\delta_2(v, c) = v'$. $M_\parallel$ will have transition rule $\delta_\parallel(\langle u, v\rangle, c) = \{\langle u', v\rangle, \langle u, v'\rangle\}$. This states that when in state $\langle u, v\rangle$ and reading c, the machine nondeterministically might be told by the Fairy Godmother that this c is in α and hence the state changes in M_1 but not in M_2, which changes the $M_\parallel$ state to $\langle u', v\rangle$. Or the machine might be told that this c is in β and hence the state changes in M_2 but not in M_1, which changes the state to $\langle u, v'\rangle$. The $M_\parallel$ start state is $\langle u_0, v_0\rangle$, where u_0 is the start state of M_1 and v_0 is that of M_2. The accept states of $M_\parallel$ are $\langle u_*, v_*\rangle$, where u_* is an accept state of M_1 and v_* is that of M_2.

(b) Let M_1 be the DFA along the left and M_2 be that along the top. The NFA $M_\parallel$ for $L_1 \parallel L_2$ will have the matrix of states as shown. Indicate the start state and the accept states and add all the transition edges.



- Answer:



9. Let L be a regular languages over an alphabet Σ. Consider the language

$$MIDTHIRDS(L) = \{y \in \Sigma^* \mid \exists x, z \in \Sigma^*,\ |x| = |y| = |z| \text{ and } xyz \in L\}$$

Like 0^n1^n, one likely would first guess that a DFA for this language would have to count the length and x, y, and z and hence this language would not be regular. But note that only y is a part of the input. Your task is to prove that $MIDTHIRDS(L)$ is also regular.

Hint: Let M be a DFA that computes L. We construct an NFA M' for $MIDTHIRDS(L)$ as follows. Imagine M' having five fingers on states of M. This will give M' states $\langle q_{\langle start,y\rangle}, q_{\langle start,z\rangle}, q_x, q_y, q_z\rangle$ where each of these q are states of M. Assuming y is a yes instance, these fingers together trace out the path p_{xyz} that the computation on M follows given input xyz.
A common phenomena of nondeterminism is that the Fairy Godmother provides you with some crucial information that alone you could not obtain and then your job at the end is to verify that what she said is actually true.

Informally, describe what each of the five fingers does as M' reads its input y.
What is the start state of M'?
Describe the edges of M', i.e. from some state $\langle q_{\langle start,y\rangle}, q_{\langle start,z\rangle}, q_x, q_y, q_z\rangle$ when reading character c_y, the Fairy Godmother can choose to transition to state $\langle q'_{\langle start,y\rangle}, q'_{\langle start,z\rangle}, q'_x, q'_y, q'_z\rangle$.
What are the accept states of M'.
How many states does M' have?
There is no need to prove your construction correct.

- Answer: Given a yes instance y, there exists strings x and z such that $|x| = |y| = |z|$ and $xyz \in L\}$. Let $p_{xyz} = p_x p_y p_z$ denote the path (a sequence of states) that the computation on M follows given input xyz. Let p_x, p_y, and p_z denote the portions of this path followed when reading the parts x, y, and z respectively. After t time steps in the computation of M' on this string y, M' will be in a state $\langle q_{\langle start,y\rangle}, q_{\langle start,z\rangle}, q_x, q_y, q_z\rangle$. Here $q_{\langle start,y\rangle}$ is the first state in the sequence of states p_y and $q_{\langle start,z\rangle}$ is the first in p_z, because at the end of the computation we will need to have remembered them. Here q_x is the t^{th} state in p_x, q_y the t^{th} in p_y, and q_z the t^{th} in p_z. In this way, the fingers at M states q_x, q_y, and q_z will simultaneously trace out these respective paths. Doing so will verify that these three paths have the same length, verifying that $|x| = |y| = |z|$. Doing so will also verify that $xyz \in L$.

  The start state of M' immediately has ϵ transitions so that the Fairy God mother can nondeterministically set the initial states $\langle q_{\langle start,y\rangle}, q_{\langle start,z\rangle}, q_y, q_x, q_z\rangle$. Here $q_{\langle start,y\rangle}$ and $q_{\langle start,z\rangle}$ are set to be beginning of p_y and p_z respectively. The initial value of q_x will be the start state of M. The initial value of q_y will be $q_{\langle start,y\rangle}$. The initial value of q_z will be $q_{\langle start,z\rangle}$.

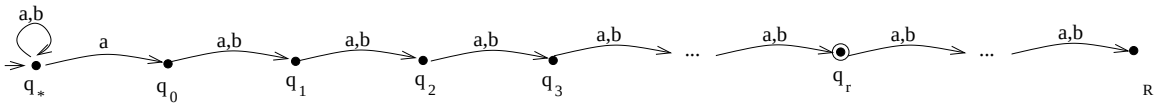
  The computation proceeds as follows. When M' reads the next character c_y of y, it transitions along the edge from its current state $\langle q_{\langle start,y\rangle}, q_{\langle start,z\rangle}, q_x, q_y, q_z\rangle$ to state $\langle q_{\langle start,y\rangle}, q_{\langle start,z\rangle}, q'_x, q'_y, q'_z\rangle$ as follows. The finger q_y transitions deterministically according to $\delta_M(q_y, c_y) = q'_y$. The Fairy God mother nondeterministically needs to tell M' the next characters c_x and c_z of x and of z. This then allows M' move the fingers q_x and q_z according to $\delta_M(q_x, c_x) = q'_x$ and $\delta_M(q_z, c_z) = q'_z$. The fingers at $q_{\langle start,y\rangle}$ and $q_{\langle start,z\rangle}$ do not move.

  When M''s input y ends, we know that x and z also end, being the same length. M' accepts if the paths p_x, p_y, and p_z connect. This requires q_x to be at state $q_{\langle start,y\rangle}$, p_y to be at $q_{start,z}$, and p_z to be at an accept state of M.

  A given y is a yes instance iff $\exists x, z \in \Sigma^*,\ |x| = |y| = |z|$ and $xyz \in L$ iff the computation on M given input xyz can be broken into three pieces $p_{xyz} = p_x p_y p_z$ iff the three paths traced by p_x, p_y, and p_z in M' match up and end at an accept state if M iff M' accepts y.

  Note that if the number of states of M is $|M|$, then the number of states of our NFA M' will have $|M|^5$ states.

10. For each integer r, consider the NFA M_r depicted below.



(a) Let the input string be of the form $\alpha = xay$, where $x, y \in \{a, b\}^*$ are strings and the specified character a is read as the computation follows this edge from q_* to q_0. What are the requirements on the substrings x and y for this α to be accepted. Namely, the language computed by M_r is $L_r = \{xay \in \{a, b\} \mid$ where ?? something about x and y ?? $\}$.

- Answer: $L_r = \{xay \in \{a, b\} \mid \text{where } |y| = r\}$.

(b) Lets index the input characters backwards, namely $\alpha = \alpha_n\alpha_{n-1}\ldots\alpha_2\alpha_1\alpha_0$. Here α_n is the first character read by the NFA and α_0 is the last. What are the requirements on the characters α_i for this α to be accepted. Namely, the language computed by M_r is $L_r = \{\alpha \in \{a, b\} \mid$ where ?? something about α_i ?? $\}$.

- Answer: $L_r = \{\alpha \in \{a, b\} \mid \alpha_r = a\}$.

(c) Using the concepts from that previous two question, explain accepting computations on this NFA M_r.

- Answer: The NFA first reads and ignores the input characters $x = \alpha_n\alpha_{n-1}\ldots\alpha_{r+2}\alpha_{r+1}$ while staying in the start state q_*. Then when the character α_r is read, the fairy god mother (nondeterministically) specifies that the computation should leave state q_* and transition to state q_0. To follow this edge, this character α_r needs to be an a. After reading i characters from y, i.e. i of the characters after α_r, the computation will be in state q_i. In our input, there are r more characters after α_r, namely those in y. Hence, the string will end at state q_r, which is the only accept state.

(d) Give a regular expression R_r representing this language L_r.

- Answer: The regular expression is $\{a, b\}^*a\{a, b\}^r$.

(e) Now focus on the NFA M_R where the accept state is q_R. Let M'_R denote the DFA obtained by converting this NFA M_R into a DFA as described in class. But recall the process of converting. After reading a string α, we put a clone on each state of M_R that the computation could be in, depending on which nondeterministic steps the computation took. Let $Q \subseteq [q_0, q_1, \ldots, q_R]$ be an arbitrary subset of these states. Describe a string denoted α_Q such that after reading it there is a clone on state q_0 and one each of the states specified in Q, but on no other states.

- Answer: We explained above how a clone can end up in state q_r if and only if $\alpha_r = a$. Hence, define $\alpha_Q \in \{a, b\}^*$ to be the string such that $\alpha_r = a$ if and only if $q_r \in Q$. It follows that on this α_Q, Q defines which states a clone can be on. Of course a clone can always also be on the start state q_*.

(f) How many states do you think the resulting DFA M'_R would have?

- Answer: It seems that the DFA will have a different state q_Q for every set $Q \subseteq [q_0, q_1, \ldots, q_R]$. There are 2^{R+1} such states.

(g) Let q_Q denote the state that the resulting DFA M'_R is in when in the NFA M_R there is a clone on state q_* and one each of the states specified in Q, but on no other states. After reading the character b, which state will M'_R be in? And after reading an a? Use $Q = \{5, 18, 21\}$ as an example.
Hint: Let $Q{+}1$ denote the set where each state in Q is incremented by one, i.e. $Q{+}1 = \{6, 19, 22\}$.

- Answer: After one time step, each clone on a state in Q will move ahead one step in the long path in M_R, i.e. this will put them in set of states denoted $Q{+}1$. When reading a b, the clone on state q^* will stay where it is. Hence the resulting state in M'_R will be $q_{Q+1} = q_{\{6,19,22\}}$. When reading an a, the clone on state q^* will clone itself. The first clone stay where it is. The second clone will transition over to state q_0. Hence the resulting state in M'_R will be $q_{\{q_0\}\cup(Q+1)} = q_{\{0,6,19,22\}}$.

(h) Now forget about the machines M_R and M'_R and let us focus on the language L_R. We will now set up the Bumping Lemma for this language. Let $S = \{\alpha_Q \mid Q \subseteq [0, 1, \ldots, R]\}$ be a set of distinguished first names. Here string α_Q is the string you defined in an earlier question. Recall the adversary chooses two different strings α_Q and $\alpha_{Q'} \in S$. Your task is to find a $\zeta \in \{a, b\}^*$ such that $L_R(\alpha_Q\zeta) \neq L_R(\alpha_{Q'}\zeta)$.

- Answer: Because the sets $Q \neq Q'$ are different, there is some index r that is in Q but not in Q_* (or visa versa). The language L_R accepts a string if the character that has R characters after it is an a. The number of characters after character α_r in α_Q is r. Let $\zeta = b^{R-r}$ so that the number of characters after character α_r in $\alpha_Q\zeta$ is $r + (R - r) = R$. Hence, the value of α_r determines whether $\alpha_Q\zeta$ is in the language or not. Because, $r \in Q$, the character α_r in α_Q is an a and hence $L_R(\alpha_Q\zeta)$ is true. Similarly, because, $r \notin Q'$, the character α_r in $\alpha_{Q'}$ is not an a and hence $L_R(\alpha_{Q'}\zeta)$ is false.

(i) What does this distinguished set S and the Bumping lemma tell us about the number of states in any DFA that solves L_R.

- Answer: The bumping lemma says that if S is a distinguished set for L_R then no DFA can solve L_R with fewer than $|S|$ states. Here $|S| = 2^{R+1}$.

(j) Can you make any conclusions from this?

- Answer: The NFA M_R has $R + 2$ states and any DFA solving L_R requires 2^{R+1} states. This proves that there can be an exponential gap between these two.
  Also the DFA M'_R built from the NFA M_R has the optimal number of states.