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

Instructor: Jeff Edmonds

Read Jeff's notes. Read the book. Go to class. Ask lots of question. Study the slides. Work hard on solving these questions on your own. Talk to your friends about it. Talk to Jeff about it. Only after this should you read the posted solutions. Study the solutions. Understand the solutions. Memorize the solutions. The questions on the tests will be different. But the answers will be surprisingly close.

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 $L(M) = L(M')$.

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

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

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

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

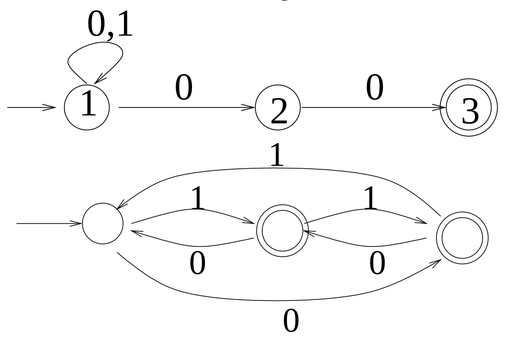
   (f) Every DFA M can be converted into an equivalent regular expression R for which $L(M) = L(R)$.

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

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

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

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

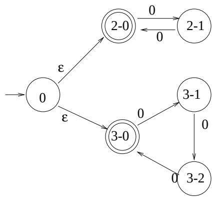


   (a) Construct an NFA for the language $L_1 \cup L_2$. Use the technique done in class that combines the above two machines. Do not simplify the machine produced.

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

   (c) 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 $L(M)$.

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

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

(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.)

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

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

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 $L(M) = \emptyset$.

   (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 $\alpha \in L(M_1) \oplus L(M_2)$.

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

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

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

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.

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,

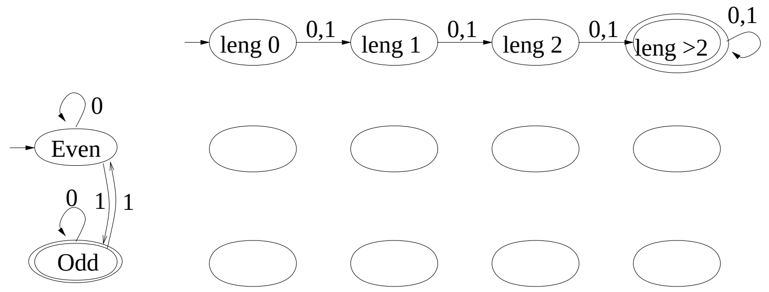
$$\{ab\} \parallel \{cd, e\} = \{abe, aeb, eab, abcd, acbd, acdb, cabd, cadb, cdab\}.$$

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 $M_{\parallel}$.

(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.



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

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

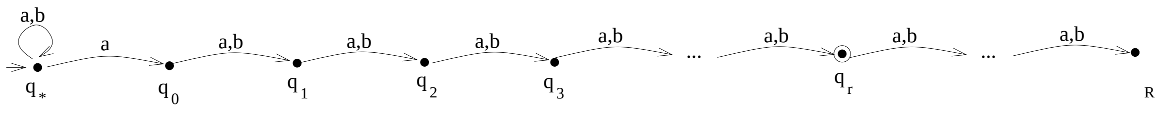
Like 0^n1^n, one likely would first guess that a DFA for this language would have to count the length and x, y, and z and hence this language would not be regular. But note that only y is a part of the input. Your task is to prove that $MIDTHIRDS(L)$ is also regular.

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 $\left\langle q'_{\langle start,y\rangle}, q'_{\langle start,z\rangle}, q'_x, q'_y, q'_z \right\rangle$.
What are the accept states of M'.

How many states does M' have?
There is no need to prove your construction correct.

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 $L_r = \{xay \in \{a,b\} \mid \text{where ?? something about } x \text{ and } y \text{ ?? }\}$.

(b) Lets index the input characters backwards, namely $\alpha = \alpha_n\alpha_{n-1}\dots\alpha_2\alpha_1\alpha_0$. 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
$L_r = \{\alpha \in \{a,b\} \mid \text{where ?? something about } \alpha_i \text{ ?? }\}$.

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

(d) Give a regular expression R_r representing this language L_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 $Q \subseteq [q_0, q_1, \dots, q_R]$ 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.

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

(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. $Q{+}1 = \{6, 19, 22\}$.

(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 $S = \{\alpha_Q \mid Q \subseteq [0, 1, \dots, R]\}$ 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 $L_R(\alpha_Q\zeta) \neq L_R(\alpha_{Q'}\zeta)$.

(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.

(j) Can you make any conclusions from this?