CSE 396 Lecture 7

Clay, James February 16, 2016

1. NFA Review

Definition 1. Nondeterministic Finite Automaton: A **nondeterministic finite automaton** is a 5-tuple $(Q, \Sigma, \delta, q_0 F)$, where

(1) Q is a finite set of states,
(2) Σ is a finite alphabet,
(3) $\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$ is the transition function,
(4) $q_0 \in Q$ is the start state, and
(5) $F \subseteq Q$ is the set of accept states

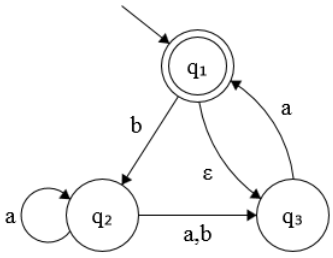

Figure 1. Figure 1.36 and 1.42 from page 53/57 (3rd edition).

What are the accept strings? Well, we know it accepts the empty string, ϵ. We see an ϵ transition that accepts a as well. What about when we have two or more a's? What about when we have a b, what strings are accepted then?

1.1. Usefulness of NFAs.

Let's come up with an NFA $L(N)$ over $\{0, 1\}$ for only the strings that contain a pair of 00 (e.g. two 00) separated by an even number of characters.

Some strings in the language might be 0100110011, 01100101100101, and 01001000. Strings not in the language are 0100100, 1011001, and 0111011.

Now let's do an NFA for all of those strings missing at least one letter from the alphabet, $\Sigma = \{a, b, c\}$.

2. NFA to DFA Algorithm

Here's the idea: keep track of the possible states an NFA can be in via a set of states (the tree diagram from 49 shows how to maintain this set based on the input). Now make each

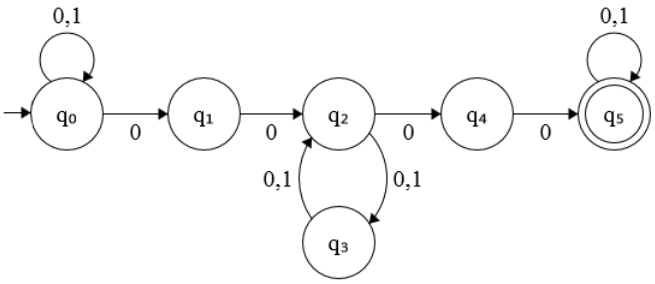

FIGURE 2. Review example without ϵ transitions. Notice that q_3 is absolutely necessary: it gobbles up an even number of characters!

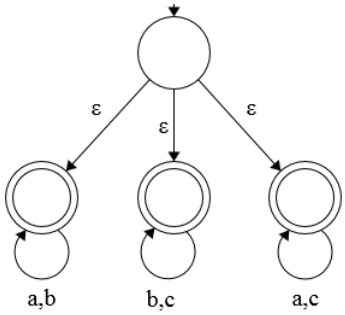

FIGURE 3. Review example for matching exactly one letter of the alphabet.

set of states from the NFA a single DFA state. So if the NFA could be in states q_1 and q_2 for a particular input, we'll make a DFA state that *combines* q_1 and q_2.

Said yet another way: this means that each state in the DFA represents a set of states from the NFA!

So we need an algorithm that takes into account the multiple state to one state conversion. We already noted that the DFA has a single state for the multiple states an NFA might have... So if the NFA has k states the DFA may need to keep track of upwards of 2^k states (all the possible subsets of the NFA states, e.g. the powerset of the NFA states).

Further, we need to determine which states will be the start and accept/finish states.

2.1. **The algorithm.** Let $N = (Q, \Sigma, \delta, q_0, F)$ be the NFA recognizing a language A. We construct a DFA $M = (Q', \Sigma, \delta', q_0', F')$ recognizing A. For the following let

$$E(R) = \{q | q \text{ can be reached from } R \text{ by traveling along 0 or more } \epsilon \text{ arrows } \}$$

(1) $Q' = \mathcal{P}(Q)$. Every state of M is a set of states of N. Recall $\mathcal{P}$ is the powerset.
(2) For each set $R \in Q'$ and $a \in \Sigma$, let $\delta'(R, a) = \{q \in Q \mid q \in E(\delta(r, a)) \text{ for some } r \in R\}$.
(3) $q_0' = E(\{q_0\})$. M starts in the state that contains the start state of N and all reachable by N via ϵ transitions.
(4) $F' = \{R \in Q' | R \text{ contains an accept state of } N\}$. So M accepts if we end up in one of the possible states that N would accept in.

2.2. **Examples.**

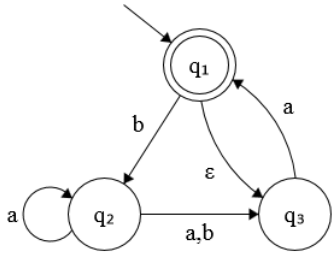

FIGURE 4. Figure 1.36 and 1.42 from page 53/57 (3rd edition).

Example 1: So doing this a step at a time:

(a) In the first step we find the powerset of all the states from the NFA.

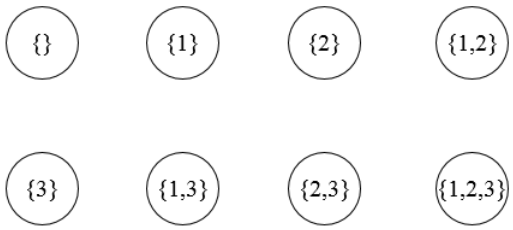

FIGURE 5. Step 1

(b) In the second step we determine the possible transitions from each set of states to another set of states (we went over this in class after doing step 3, e.g. finding the start state, but it can be done in either order).

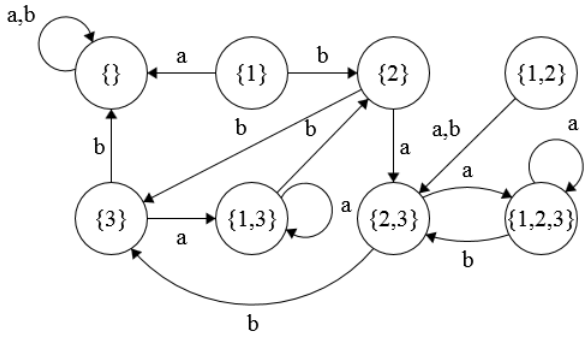

FIGURE 6. Step 2

Going clockwise starting from the empty set we can fill out the diagram. Here we will only do the top row (we did the entire diagram in class), but if you want further help the book goes over this in detail on pages 57 and 58.

- Where will the empty set {} or $\emptyset$ go on any input? Formally a state in the NFA transitions to the $\emptyset$ if no valid transitions exist given the current input. So that instance of the NFA "dies" in some sense. Thus any input after the "death" will force that instance to remain "dead" and thus we return to the $\emptyset$ state for all inputs.
- Now suppose we're in $\{1\}$. So if we get an 'a' as input we have the following $\delta'(R, x)$ transition:

  $$\delta'(\{1\}, \text{'a'}) = \{q \in Q | q \in E(\delta(r, \text{'a'})) \text{ for some } r \in \{1\}\}$$

  which begs the question, what is $E(\delta(r, \text{'a'}))$? Well we have only a single element in $R = \{1\}$ so we only have to worry about $E(\delta(1,' a'))$. So first resolving the inner δ (recall this δ is from the NFA): $\delta(1, \text{'a'}) = \emptyset$. Why? Well from the NFA diagram we can see that if we're in q_1 (or just state 1 for our purposes) there is no transition with just an 'a'. Even though we have a ϵ transition, this occurs AFTER we've read and transitioned based on the current state. So if we read an 'a' in state $\{1\}$ we transition to the empty state.
  On the other hand if we get a 'b', then $E(\delta(\{1\}, \text{'b'})) = \{2\}$ (verify this!).
- Suppose we're in state $\{2\}$ now. If we get an 'a' we can either stay in state 2 or we can go to state 3. So our transition is to state $\{2, 3\}$ (because both are valid in the NFA). If we get a 'b' however, then state 2 must transfer to state 3.
- Suppose we're in state $\{1, 2\}$. What does that mean? Well it means that we're supposing both states are potentially active in the NFA (remember the NFA can be in multiple states at the same time in some sense). Now suppose we get an 'a' in and we're in state 1, well as we saw previously, this means we transition to the empty set state. But what if we get an 'a' in state 2? Well here we have two options. We can stay in state 2 or we can go to state 3. So $\{2, 3\}$ is the result of being in state $\{2\}$ and reading an 'a'.
  What about 'b'? Well, if we're in state 1 and we read a 'b' we go to state 2. If we're in state 2 and we read a 'b' we go to state 3. So we really transitioned to either 2 or 3, thus state $\{1, 2\}$ also transitions to state $\{2, 3\}$.

(c) Now we need to determine the start state. We know that (1) could be a start state. But we also have the ϵ arrow from (1) to (3). So 3 could be the start state as well. Thus we choose the start state as the set of states that contain exactly (e.g. only) 1 and 3.

(d) To finish we need to determine which states are the accept states. This is actually pretty straightforward: just look for any set of states in the DFA that contains an accept state from the NFA. As the NFA above had state 1 as the only accept state you just need to look for states in the DFA that contain 1. So the following are all accept states: $\{1\}, \{1, 2\}, \{1, 3\}, \{1, 2, 3\}$.

Notice that not all states are reachable and we can redraw this without the extra states (a breadth first search from the start state $\{1, 3\}$ shows which states are reachable).

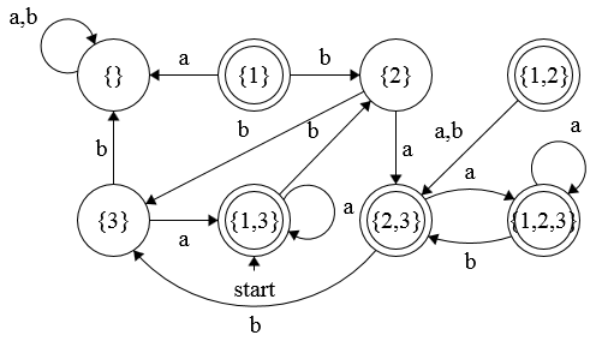

FIGURE 7. Steps 3 and 4

Example 2: We'll go over the second NFA example with ϵ transitions but restrict it to accept only those strings that don't contain both 'a' and 'c' in addition to those that are missing at least one of the alphabet. (Can you tell which states are unreachable?)

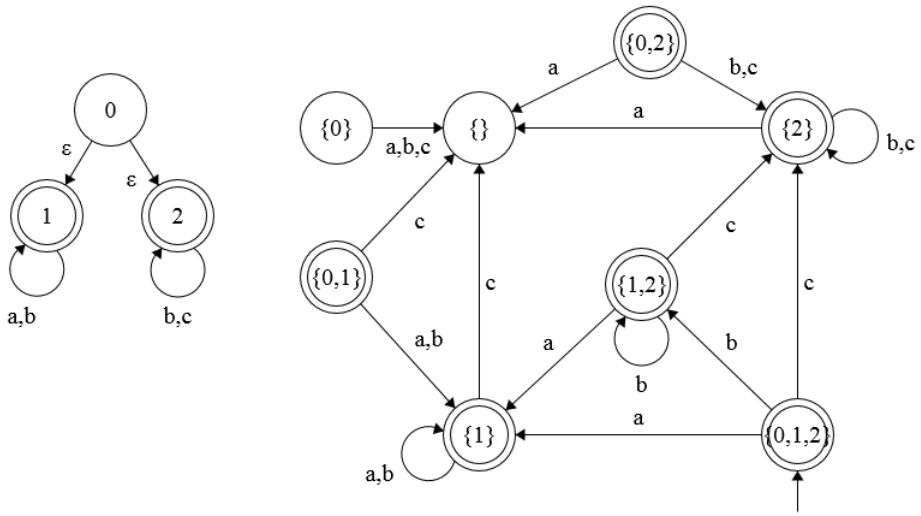

FIGURE 8. The paired down NFA from example 2