

COMP 481: Automata, Formal Languages, and Computability
 Spring 2009
 Solutions to Homework Assignment #2

1. (a) An NFA for the language is shown in Figure 1.

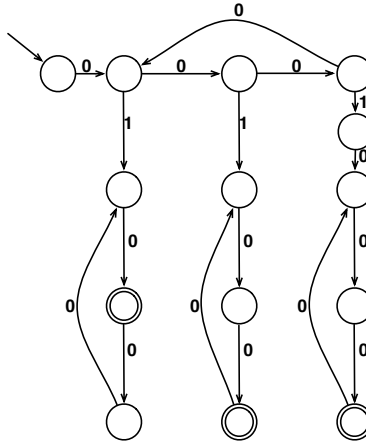


Figure 1: NFA for the language in Problem 1a.

- (b) An NFA for the language is $M = (K, \Sigma, \Delta, s, A)$, where

- $K = \{s, q_f\} \cup \{q_i : 1 \leq i \leq n\}$
- $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_n\}$
- $\Delta =$
 - $\{(s, \sigma_i, q_i) : 1 \leq i \leq n\}$ [First character is σ_i]
 - $\cup \{(q_i, \sigma_i, q_f) : 1 \leq i \leq n\}$ [Last character is σ_i]
 - $\cup \{(s, \sigma_i, q_f) : 1 \leq i \leq n\}$ [The string has only one character]
 - $\cup \{(q_i, \sigma_j, q_i) : 1 \leq i \leq n, 1 \leq j \leq n\}$ [Consume all characters between the first and last]
- $A = \{s, q_f\}$

2. The first step is to compute the ε -closure of each state, which we denote by $E(q)$, for state q :
 $E(q_0) = \{q_0, q_1, q_3\}$, $E(q_1) = \{q_1, q_3\}$, $E(q_2) = \{q_2\}$, $E(q_3) = \{q_3\}$, $E(q_4) = \{q_2, q_4, q_5\}$,
 and $E(q_5) = \{q_5\}$.

Then, we start “determinizing” the NFA, starting from $E(q_0)$. The resulting DFA has 4 states and 8 transitions, as follows:

- $K = \{q_{013}, q_{245}, q_{135}, q_\emptyset\}$
- $\delta = \{(q_{013}, 0, q_{245}), (q_{013}, 1, q_{013}), (q_{245}, 0, q_{245}), (q_{245}, 1, q_{135}), (q_{135}, 0, q_{245}), (q_{135}, 1, q_\emptyset), (q_\emptyset, 0, q_\emptyset), (q_\emptyset, 1, q_\emptyset)\}$.

- $s = q_{013}$
 - $A = \{q_{245}, q_{135}\}$
3. (a) Let $M = (K, \Sigma = \{0\}, \delta, s, A)$ be a DFA for language L . We construct an NFA $M' = (A', \Sigma' = \{a, b\}, \delta', s', A')$ which recognizes $\text{SplitP}(L)$. The idea is as follows: we create two copies M_1 and M_2 of DFA M , with M_1 having transitions on symbols a only, and M_2 having transitions on symbol b only. Then, from each state in M_1 , we have an ε -transition to its “copy” in M_2 . The start state of M is the start state in M_1 , and the accepting states are all the accepting states of M_1 and M_2 . This way, for string 0^p accepted by M , we now have a^r accepted traversed by M_1 , and b^s traversed by M_2 , where $r + s = p$, and $r, s \geq 0$. Formally,
- $K' = \{q^1, q^2 : q \in K\}$.
 - $\delta'(q^1, a) = p^1$, where $\delta(q, 0) = p$, for $p, q \in K$ and $p^1, q^1 \in K'$,
 - $\delta'(q^2, b) = p^2$, where $\delta(q, 0) = p$, for $p, q \in K$ and $p^2, q^2 \in K'$,
 - $\delta'(q^1, \varepsilon) = q^2$, for $q^1, q^2 \in K'$.
 - $s' = s^1$.
 - $F' = \{q^1, q^2 : q \in A\}$.
- (b) The idea for proving that $A_{\frac{1}{2}-}$ is regular for a regular language A is:
- i. Guess a final state $q \in F$ and a y such that $|x| = |y|$,
 - ii. Simulate M “in parallel” on x from s forward and on y from q backwards, and
 - iii. Accept if both simulations end at the same state.
- Formally, let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA such that $L(M) = A$. We construct FA $M' = (Q', \Sigma, \delta', q'_0, F')$ such that $L(M') = A_{\frac{1}{2}-}$, where
- $Q' = (Q \times Q) \cup \{q'_0\}$.
 - $\delta'(q'_0, \varepsilon) = \{[q_0, q_j] : q_j \in F\}$
 - $\delta'([q_i, q_j], a) = \{[q'_i, q'_j] : q'_i = \delta(q_i, a) \text{ and } q'_j = \delta(q_j, b) \text{ for some } b \in \Sigma\}$.
 - $F' = \{[q, q] : q \in Q\}$.
4. A DFA $M_n = (K_n, \Sigma, \delta_n, s_n, A_n)$ that accepts language C_n is the following:
- $K_n = \{k_0, k_1, \dots, k_{n-1}\}$, where state k_i , $0 \leq i \leq n-1$, corresponds to binary numbers x such that $x \equiv_n i$.
 - $\Sigma = \{0, 1\}$.
 - $\delta_n(k_i, \sigma) = ((2k_i + \sigma) \bmod n)$ for every $k_i \in K$ and every $\sigma \in \Sigma$.
 - $s_n = k_0$.
 - $A = \{k_0\}$.
5. (a) The set of all strings in the language a, b in which the m th to last character is an a .
- (b) • The first state accepts all strings going back to itself, or an a transition to the second state.

- The $m - 1$ middle states accept either an a, b transition to the next state only.
- The final state is an accept state with no outbound transition.

The above NFA will accept a string when it waits for the m to last state to arrive, notes that it is an a by transitioning to the second state, and then makes $m - 1$ more transitions to a final accept state.

The above NFA can only reach the accept state when it sees an a and goes through $m - 1$ transitions. Any further transitions would 'crash' the NFA, as it cannot move past the accept state.

- (c) We create 2^m states labeled with all m -length strings in this alphabet. We start in the state with m b 's. We accept in any state in which the first character is an a . Transitions between states are as follows: take the label. Chop off the first character. Append the character you are transitioning on. Transition to the state with that label. Thus the transition from $aabb$ on a is to state $abba$.
- (d) The intuition is that we need to care about the last m characters, each representing a bit of data. We need a state for every possible combination of these last characters, and there are 2^m possible combinations.

To be more formal: recall that two strings a and b can be distinguished by noting that there is another string x such that ax is in a language and bx is not. These strings must transition to different states in any DFA that accepts the language.

Note that for any two strings with a different character in any of the last m bits string, we can distinguish the strings with respect to l as follows.

- Let the character that is different be in location k .
- Note that one string must have an a at character k , and the other must have a b . Name these the a - string and the b - string, respectively.
- Let $n - m + 1$ be j .
- Note that $k \geq j$, as the character at x j resides in the last m bits by definition.
- Let x be the string of $k - j$ b 's to the string.
- Note that the a - string appended by x has an a in the $n - m + 1$ th position, while b - string appended with x does not.
- Thus a - string x is in L , but b - string x is not. a - string and b - string must be represented by two different states.

Thus any two strings that are different in the last $n - m + 1$ bits must be in different states. This is say that the last m characters in a string define a class which is, at minimum, distinct from any other m character ending. Thus there are at least as many different states as m -character strings in this alphabet. And there are, of course, 2^m such strings.

6. (a) Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA such that $L(M) = L$. We construct an FA, $M' = (Q', \Delta, \delta', q'_0, F')$ such that $L(M') = h(L)$. Let $k = \max_{\sigma \in \Sigma} |h(\sigma)|$.
- $Q' = Q \times \{w : w \in \Delta^*, |w| \leq k\}$.

- $q'_0 = [q_0, \varepsilon]$.
 - $F' = F \times \{\varepsilon\}$.
 - For every $\delta(p, a) = q$, ($p, q \in Q$, $a \in \Sigma$), introduce the following transitions to δ' :
 - $\delta'([p, \varepsilon], \varepsilon) = [p, h(a)]$.
 - $\delta'([p, \sigma w], \sigma) = [p, w]$ for every $\sigma \in \Delta$ and $w \in \Delta^+$.
 - $\delta'([p, \sigma], \sigma) = [q, \varepsilon]$.
- (b) Let $M = (Q, \Delta, \delta, q_0, F)$ be a DFA such that $L(M) = L$. We construct an FA $M' = (Q, \Sigma, \delta', q_0, F)$ such that $L(M') = \{w \in \Sigma^* : h(w) \in L\}$. Define δ' as follows:
- $\delta'(q, a) = \delta^*(q, h(a))$ for every $q \in Q$ and $a \in \Sigma$.