CSE 547T Automata Theory January 24, 2017

Homework 1 Practice Problem Solutions

WARNING: if you haven't at least tried hard to solve the practice problems before reading these solutions, you are missing the point. If you can't make *any* progress, talk to me or to the TAs before reading these solutions. Otherwise, you should come up with a solution of your own that you can compare to the one shown here.

1. To prove the first property, we reason from the definitions as follows:

$$\begin{aligned} \delta^*(\psi, \varepsilon) &= \bigcup_{q \in \psi} \delta^*(q, \varepsilon) \\ &= \bigcup_{q \in \psi} \{q\} \\ &= \psi. \end{aligned}$$

   To prove the second property, proceed inductively on $|x|$ (here, x is nonempty). In the base case, $x = a$ (i.e. $y = \varepsilon$), and we have

$$\begin{aligned} \delta^*(\psi, a) &= \delta(\psi, a) \\ &= \delta(\delta^*(\psi, \varepsilon), a). \end{aligned}$$

   In the general case, we have

$$\begin{aligned} \delta^*(\psi, ya) &= \bigcup_{q \in \psi} \delta^*(q, ya) \\ &= \bigcup_{q \in \psi} \bigcup_{r \in \delta^*(q,y)} \delta(r, a) \\ &= \bigcup_{r \in \bigcup_{q \in \psi} \delta^*(q,y)} \delta(r, a) \\ &= \bigcup_{r \in \delta^*(\psi,y)} \delta(r, a) \\ &= \delta(\delta^*(\psi, y), a), \end{aligned}$$

   where the fourth step follows by the inductive hypothesis because $|y| < |x|$.

2. (Note: reading the input left-to-right is not much harder than reading it right to left; in fact, it gives a smaller DFA. I'll leave that construction as an exercise.)

   First, let's reduce this problem to something that DFAs are very good at: counting in modular arithmetic. If the problem were to identify strings whose *length* is zero modulo three, that would be very easy – a cycle of three states would suffice. Equivalently, we can say that it would be easy to solve the problem if the input number were in *unary* notation (n represented by the string 0^n). Our machine will "translate" its input from binary to unary, then use the simple three-state cycle to track the input value modulo three.

To perform the translation, we first observe the following fact:

$$2^k \bmod 3 = \begin{cases} 1 & \text{if } k \text{ is even} \\ 2 & \text{if } k \text{ is odd.} \end{cases}$$

This fact is easily proven by induction on k. When $k = 0$, $2^k = 1$. For larger k, observe that $2^k = 2 \times 2^{k-1}$. If $k-1$ is even, then $2^{k-1} \bmod 3 = 1$ by the inductive hypothesis, and hence $2 \times 2^k \bmod 3 = 2$. Otherwise, $2^{k-1} \bmod 3 = 2$ by the inductive hypothesis, and hence $2 \times 2^k \bmod 3 = 1$.

Let M be the DFA to accept the desired language L. We can write any binary number as $\sum_i 2^{k_i}$, where the i's correspond to the positions of the 1 bits in the number. For example, 12 is decomposed as $1100 = 2^4 + 2^3$. Now M can certainly track whether it has seen an odd or even number of bits. Hence, it can tell whether each "1" bit in the input adds 1 or 2 to the value of the number modulo 3.

Rather than draw a picture of the automaton M, let's describe it in tabular form. There are 7 states divided into three groups. The first three "even" states, q_{e0} through q_{e2}, encode the correct transitions when the next bit read would be 2^k for even k. The next three "odd" states, q_{o0} through q_{o2}, encode the same information for odd k. The remaining state, q_x, is the initial state, which must be distinguished because M cannot accept the empty string. Only states q_{e0} and q_{o0} are accepting. Here is δ:

	0	1
q_{e0}	q_{o0}	q_{o1}
q_{e1}	q_{o1}	q_{o2}
q_{e2}	q_{o2}	q_{o0}
q_{o0}	q_{e0}	q_{e2}
q_{o1}	q_{e1}	q_{e0}
q_{o2}	q_{e2}	q_{e1}
q_x	q_{o0}	q_{o1}

The key property to prove is this: if M has thus far read an input string encoding a binary number n, it will be in state q_{pv}, where $v = n \bmod 3$ and p is the parity of the number of bits read (**e**ven or **o**dd). If we can prove this property, it follows that M accepts iff $n \bmod 3 = 0$, since M accepts only in states q_{e0} and q_{o0}. We proceed by induction on the number of bits read.

After reading 1 bit b, we have by construction that M is left in state q_{ob}, as desired. In the general case, suppose the input is of the form $n = b \cdot m$, where b is the high-order bit and m is the rest of the number. By induction, reading m leaves M in state $q_{p'v'}$, where $v' = m \bmod 3$ and p' is the parity of the number of bits in m. Consider what happens when we read b. First, all transitions go from even to odd or odd to even, so p is the opposite of p'. Second, if $b = 0$, inspection shows that $v' = v$. Finally, if $b = 1$, and that "1" represents 2^k, observe that p' is odd iff k is odd. Inspection of the transitions shows that $v = (v' + 1) \bmod 3$ when p' is even, and that $v = (v' + 2) \bmod 3$ when p' is odd, as desired. Conclude that the state q_{pv} satisfies the claimed property for n.

3. If $M = (Q, \Sigma, q_0, A, \delta)$, define $\overline{M} = (Q, \Sigma, q_0, \overline{A}, \delta)$. Every item except the accepting set $\overline{A}$ is the same as the corresponding item for M. We define the accepting set for $\overline{M}$ to be $\overline{A} = Q - A$.

   Because δ is the same for both M and $\overline{M}$, so too is δ^*. For a string x, let $q = \delta^*(q_0, x)$. By construction, $q \in A$ iff $q \notin A'$. Conclude that M accepts precisely when $\overline{M}$ does not.

4. **(a)** We want to show that M_k accepts precisely strings yz ending with a suffix of the form $z = 1\{0,1\}^k 1$. Clearly, there is an accepting computation for all such strings: remain in state q_0 until

y is consumed, then traverse the remaining states to read z. To argue the other direction, observe that any accepting computation must end with the series of states $q_0, s_0, s_1, \ldots s_k, q_a$, since there is nowhere to go from q_a. This path corresponds to a string of the form $1\{0,1\}^k 1$.

(b) The lazy subset construction has the nice property that it only constructs DFA states corresponding to sets of NFA states that are reachable by *some* string. Hence, we need to show that there are exponentially many such reachable sets for M_k.

Consider a $k+1$-bit string t of the form $b_k b_{k-1} b_{k-2} \ldots b_0$. What set of states is M_k in after reading t? Observe that M_k can only transition from q_0 to s_0 by reading a "1" bit. Moreover, once this transition occurs, the next j bits read lead only to state s_j. These observations imply the following fact about M_k:

For $0 \leq j \leq k$, the set $\delta^*(q_0, t)$ includes state s_j iff $b_j = 1$.

Indeed, M_k has a path to s_j on t iff it can take the transition $q_0 \to s_0$ with exactly j bits remaining, which is true iff $b_j = 1$.

Now every state set ψ of M_k reachable on some string t of length $k+1$ must contain q_0, since M_k may simply loop in q_0 until the whole string is read. By setting some subset of bits $b_{i_1}, b_{i_2}, \ldots b_{i_m}$ of t to 1, we can additionally add exactly the states $s_{i_1}, s_{i_2}, \ldots s_{i_m}$ to ψ. There are 2^{k+1} ways of choosing which bits of t will be 1's, so there are 2^{k+1} distinct sets ψ reachable from q_0 on strings of length $k+1$. Conclude that the lazy subset construction for M_k will yield a DFA with at least 2^{k+1} states. The DFA cannot have more than 2^{k+3} states, since there are only $k+3$ states in the entire NFA, so the DFA will indeed have $\Theta(2^k)$ states.