CS500 Homework #1 Solutions

1. (Exercise 1.4 c, d, and e, and Exercise 1.13) Draw diagrams of DFAs that recognize the following languages, and give regular expressions for each one.

   (a) $\{w \mid w$ contains the substring 0101$\}$

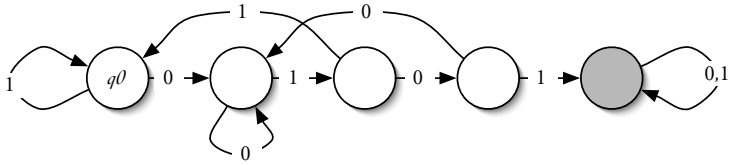

Figure 1: $(0+1)^*0101(0+1)^*$. Gray states are accepting.

   (b) $\{w \mid w$ has length at least 3 and its third symbol is 0$\}$

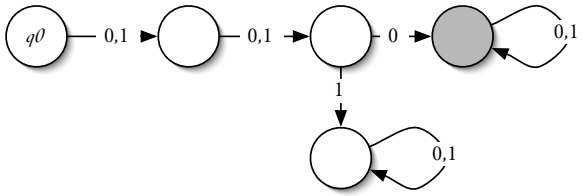

Figure 2: $(0+1)(0+1)0(0+1)^*$. Gray states are accepting.

   (c) $\{w \mid w$ starts with 0 and has odd length, or starts with 1 and has even length$\}$

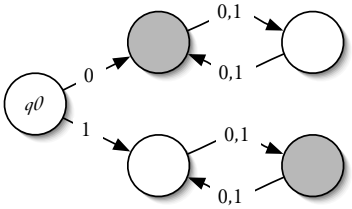

Figure 3: $0((0+1)(0+1))^* + 1(0+1)((0+1)(0+1))^*$. Gray states are accepting.

2. Recall the definition $\delta^*(q, w_1 w_2 \cdots w_n) = \delta^*(\delta(q, w_1), w_2 \cdots w_n)$ for the effect on a DFA of reading a word w. The language L recognized by a machine can then be written

$$L = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}$$

where q_0 is the initial state and F is the set of accepting states.

Given a DFA M, we can define an equivalence $\sim_M$ on words $u, v \in \Sigma^*$ as follows:

$$u \sim_M v \text{ if and only if } \delta^*(q_0, u) = \delta^*(q_0, v)$$

and, as we discussed in class, given a language L we can define an equivalence $\sim_L$ as

$$u \sim_L v \text{ if and only if } (\forall w : uw \in L \Leftrightarrow vw \in L)$$

Prove formally that if M recognizes L, then $u \sim_M v$ implies $u \sim_L v$. Explain why the converse might not be true—that is, why $\sim_M$ might be a "finer" equivalence, dividing Σ^* into more equivalence classes, than $\sim_L$.

Answer. Suppose $u \sim_M v$, so $\delta^*(q_0, u) = \delta^*(q_0, v)$. By induction, it is easy to show that $\delta^*(q, xy) = \delta^*(\delta^*(q, x), y)$ for any state q and any strings x and y. Therefore, for any string w,

$$\delta^*(q_0, uw) = \delta^*(\delta^*(q_0, u), w) = \delta^*(\delta^*(q_0, v), w) = \delta^*(q_0, vw)$$

Thus $\delta^*(q_0, uw) \in F$ if and only if $\delta^*(q_0, vw) \in F$, and if M recognizes L this means that $uw \in L$ if and only if $vw \in L$. Thus $u \sim_L v$. □

The converse is not true, since if M is not the minimal DFA that recognizes L, it could send u and v to separate states even though $u \sim_L v$. In that case M has more states, and $\sim_M$ divides Σ^* into smaller pieces than $\sim_L$.

3. Using the same notation as in the previous problem, prove that if $u \sim_L v$, then for any $a \in \Sigma$ we have $ua \sim_L va$. Show that this allows us to can define a DFA M whose set of states Q is the set of equivalence classes of $\sim_L$; define q_0, δ and F, and prove that M recognizes L. Moreover, prove that M is the smallest DFA that recognizes L, and that it is (up to isomorphism) the only DFA of this size that recognizes L.

Answer. Suppose $u \sim_L v$; then for all w, $uw \in L$ if and only if $vw \in L$. This includes words w which begin with a, i.e., words of the form $w = ax$. Thus, for all x, $uax \in L$ if and only if $vax \in L$, so $ua \sim_L va$. □

This has the following consequence: if I ask you for which equivalence class ua is in, you don't need to know u; you only need to know u's equivalence class. Therefore, if q is an equivalence class, we can define $\delta(q, a)$ as the equivalence class containing ua where u is any word in q. (If we didn't have the above result, there might be several such equivalence classes, in which case $\delta(q, a)$ would not be well-defined.) Similarly, we let q_0 be the equivalence class containing the empty word ϵ, and let F be the set of equivalence classes containing words in L. (Note that the words in an equivalence class are either all in L or all not in L; to see this, set $w = \epsilon$.)

Once we set all this up, the state $\delta^*(q_0, w)$ is simply the equivalence class containing w, which is in F if and only if $w \in L$; so M recognizes L.

Moreover, the result of #2 above shows that the number of states of any machine M is at least the number of equivalence classes of $\sim_L$. Therefore, if M is minimal, the equivalence classes of $\sim_M$ are *exactly* the equivalence classes of $\sim_L$; then there is a one-to-one correspondence between the states of M and equivalence classes, and any such machine is isomorphic to the one described here. □

4. What are the equivalence classes of $\sim_L$ if $L = \{w \in \{0,1\}^* | w \text{ contains the substring } 101\}$? Use this to draw the minimal DFA for L.

   Answer. There are four equivalence classes:

   (a) w does not contain 101, and does not end in 1 or 10 (this is the start state)

   (b) w does not contain 101, but ends in 1

   (c) w does not contain 101, but ends in 10

   (d) w contains 101 (the accepting state).

   Considering the transitions between these gives the following DFA:

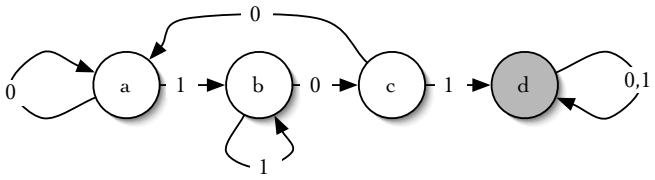

Figure 4: The minimal DFA for $(0+1)^*101(0+1)^*$.

5. (Exercise 1.17c, modified) Give two proofs, one based on the pumping lemma and one based on equivalence classes, that the language $\{a^{2^n} \mid n \in \mathbb{N}\} = \{a, aa, aaaa, aaaaaaaa, \ldots\}$ is not regular.

   Proof 1. Suppose the pumping lemma holds for L. Then there would be a p such that for all $w \in L$ with $|w| \geq p$, there would be some way to write $w = xyz$ with $|y| \leq p$ such that $xy^iz \in L$ for all $i \geq 0$. For a unary language, $w = a^\ell$ and $y = a^k$ for some $k < p$, so $xy^iz = w^{\ell+(i-1)k}$. Then the pumping lemma becomes the claim that for all $\ell \geq p$, if $a^\ell \in L$, then $a^{\ell+(i-1)k} \in L$ for all $i \geq 0$, for some $k \leq p$. In particular, considering xy^2z, the pumping lemma claims that $a^{\ell+k} \in L$ for some $k \leq p$.

   Now suppose that $\ell \geq p$ and $a^\ell \in L$, and let a^m be the next shortest word in L. For the pumping lemma to be true, we need the gap in their lengths $m - \ell$ to be less than or equal to p; otherwise $xy^2z = a^{\ell+k}$ cannot be in the language since $\ell + k \leq \ell + p < m$. Therefore, if the gaps in a unary language don't have a constant upper bound — that is, if it has arbitrarily large gaps — it cannot be regular.

For $\{a^{2^n}\}$ in particular, if the adversary claims that the pumping lemma is true with some value of p, let w be a^ℓ where ℓ is the smallest power of 2 greater than p. Then the length gap between w and the next shortest word $a^{2\ell}$ is $\ell > p$. □

Proof 2. I claim that none of the words in L are equivalent to each other. To see this, let $u = a^{2^m}$ and $v = a^{2^n}$ for any $m \neq n$, and let $w = u$. Now $uw = a^{2 \cdot 2^m} = a^{2^{m+1}} \in L$, but $vw = a^{2^m + 2^n} \notin L$ since the sum of two distinct powers of 2 is never a power of 2. Thus $u \not\sim_L v$ for any distinct $u, v \in L$, so there are an infinite number of equivalence classes of $\sim_L$ and L is not regular. □

6. (Problem 1.30) Show that for all p, the language

$$L_p = \{w \in \{0,1\}^* \mid w \text{ represents a binary integer divisible by } p\}$$

is regular.

Answer. Let x be the number represented by the digits we've read so far. Reading the next binary digit $a = 0$ or 1 changes x to $2x + a$. To tell whether w is a multiple of p, it suffices to know $w \bmod p$, and thus to keep track of $x \bmod p$. Now, for any a, $(2x + a) \bmod p$ is a function of $x \bmod p$; this is obvious if you're familiar with modular arithmetic — if you're not, consider

$$(2x + a) \bmod p = (2(x \bmod p) + a) \bmod p \ .$$

Therefore, L_p can be recognized by a DFA with p states, one for each possible value of $x \bmod p$, where $0 \bmod p$ is the accepting state. (The one problem with this machine is that it accepts the empty word, but presumably ϵ represents zero.) Note that a similar construction works for numbers in any base; there's nothing special about binary. □

7. (Problem 1.39) Prove that for all $k > 1$, DFAs with k states are more powerful than DFAs with $k - 1$ states. That is, come up with a family of languages L_k such that L_k can be recognized by a DFA with k states but not by a DFA with $k - 1$ states. Thus the DFAs form a strict hierarchy based on the number of states, which classify the regular languages according to their complexity.

Answer. The languages L_p from the previous question require p states whenever p is odd, but here's a simpler example: let L_k be the set of words of length at least $k - 1$. It's clear that L_k has k equivalence classes, corresponding to words of length $0, 1, 2, \ldots, k - 2$ and $k - 1$ or more. Thus L_k requires a DFA with k states. □

8. (Problem 1.44 — a little harder) Now prove that NFAs can be exponentially more compact than DFAs, i.e., that the exponential blowup in the number of states we get when converting an NFA to a DFA is sometimes necessary. That is, give a family of languages L_k such that L_k can be recognized by an NFA with k states, but whose minimal DFA has $\Omega(2^k)$ states. Hint: Look at Example 1.14 in Sipser.

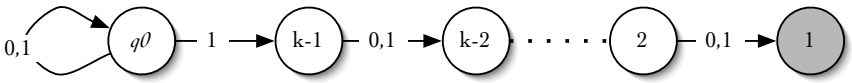

Figure 5: The NFA for L_k.

Answer. Let L_k be the language of words in $\{0,1\}^*$ where the $(k-1)$th-to-last symbol is a 1. Example 1.14 is L_4. L_k can be recognized by an NFA with k states; the start state q_0, and $k-1$ states labeled $k-1, k-2, \ldots, 1$. The NFA is shown in the figure. We can stay in the start state as long as we want, but when we see a 1 we can guess that it's the $(k-1)$th-to-last symbol, and then count down from there. If we were right, we end in the accepting state at the end of the word.

But, the minimal DFA for L_k has at least 2^{k-1} states. To see this, suppose that u and v differ somewhere in their last $k-1$ symbols. In that case, there is some $j < k$ such that the jth-to-last symbol of u is 1 but the jth-to-last symbol of v is 0 (or vice versa). Then, if w is of length $j-1$, $uw \in L$ but $vw \notin L$, so $u \not\sim v$. Therefore, there are at least 2^{k-1} equivalence classes, one for each final sequence of $k-1$ symbols. □

9. (Problem 1.24 plus...) Given a word $w = w_1 \cdots w_n$, its *reverse* is $w^R = w_n \cdots w_1$, and the reverse of a language L is $L^R = \{w^R \mid w \in L\}$. Show that if L is regular, so is L^R. However, show also that the number of states for the minimal DFA for L^R can be exponentially larger than the number for L. Hint: use the same languages as in the previous problem.

   Answer. If L can be written as a regular expression, then so can L^R. Since a regular expression is built from smaller ones using union, concatenation, and the Kleene star, we can prove this by induction on the size of the expression as follows:

$$\begin{aligned} (L_1 \cup L_2)^R &= L_1^R \cup L_2^R \\ (L_1 L_2)^R &= L_2^R L_1^R \\ (L_1^*)^R &= (L_1^R)^* \end{aligned}$$

   The base case is given by the fact that $a^R = a$ for a single symbol a. □

   In the previous problem we showed that L_k requires a DFA of $\Omega(2^k)$ states. However, L_k^R can be recognized with a DFA of only k states: simply reverse the NFA shown above and confirm that the $(k-1)$th symbol is a 1. Thus $L_k = (L_k^R)^R$ requires exponentially more states than L_k^R does.

10. (Problem 1.41) Prove that the language

$$L = \{w \mid w \text{ has an equal number of 01s and 10s}\}$$

    is regular.

Answer. This is kind of a trick question — L is simply the language of words which begin and end with the same symbol, i.e.,

$$0(0+1)^*0 \ + \ 1(0+1)^*1 \ .$$

To see this, note that we get a 01 or 10 each time we switch from 0 to 1 or back again, and if we start and end with the same symbol we switch back and forth an equal number of times. □

11. (Problem 1.42 — Tricky!) For a language L, define $L_{1/2}$ as

$$L_{1/2} = \{x \mid xy \in L \text{ for some word } y \text{ such that } |y| = |x|\}$$

That is, $L_{1/2}$ is the set of "first halves" of L, namely the set of words x that can be followed by words y of the same length giving a word xy in L. Prove that if L is regular, so is $L_{1/2}$.

Answer. We can't say things like "accept at the half-way point," since there is no way for the DFA to know when it is half-way through the input. What we have to do is simulate two copies of the DFA in parallel; one moves forwards from the start state by reading x, while the other non-deterministically moves backwards from an accepting state by guessing y. Then if both states are equal, this means that we met in the middle and $xy \in L$.

Formally, let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA for L, and define a new machine, an NFA, $M' = (Q', \Sigma, \delta', q_0', F')$ as follows. Let $Q' = Q \times Q$ corresponding to the two copies of M. Let us expand our definition of an NFA slightly by letting it non-deterministically start in any element of a starting set $Q_0' = \{(q_0, q_f) \mid q_f \in F\}$; equivalently, let its start state be q_0' and allow ϵ-transitions from q_0' to each $q' \in Q_0'$. Then define

$$\delta'((q_1, q_2), a) = \{(\delta(q_1, a), q_3) \mid \exists b : \delta(q_3, b) = q_2\}$$

Note that b is the next symbol of y, read in reverse, which we guess to move backwards from q_2 to q_3. Finally, define $F' = \{(q, q)\}$. Then M' recognizes $L_{1/2}$. □

You can generalize this in various ways; for instance, you can define L_{mid} as

$$L_{\text{mid}} = \{y \mid \exists x, z : |x| = |y| = |z| \text{ and } xyz \in L\} \ .$$

Then L_{mid} is the set of middle thirds of words in L, and by guessing the first and last thirds x and z we can prove that L_{mid} is regular if L is. (The converse is not true.)

12. Show that

$$L = \{w \in \{0,1\}^* \mid w \text{ represents a prime number in binary}\}$$

is not regular. You may assume the following conjecture: there are an infinite number of *Mersenne primes*, i.e., primes of the form $2^k - 1$. You may also use the following fact: if $2^k - 1$ is prime, then k is prime (the converse is not true). Finally, you may also use the Prime Number Theorem, which states that for some constant C, the number $\pi(x)$ of primes less than x obeys $\pi(x) < Cx/\ln x$.

Answer. Let's transform L into a unary language. Let $L' = L \cap 1^*$; since 1^k in binary represents $2^k - 1$, we have

$$L' = \{w \mid w \text{ represents a Mersenne prime}\}$$

If we can prove that L' is not regular, then since the set of regular languages is closed under intersection it follows that L is not regular.

As in question #5, it's enough to show that the set of lengths of words in L' has arbitrarily large gaps. (Since any finite language is regular, we also need to know that L' is infinite; for this we rely on the conjecture that there are an infinite number of Mersenne primes.) First, since k must be prime for $2^k - 1$ to be prime, we have

$$L' \subset \{1^k \mid k \text{ is prime}\}$$

If the Pumping Lemma were true the gaps between adjacent primes would be bounded above by some constant p. But then the number of primes less than x would obey $\pi(x) \geq x/p$, and this would contradict the Prime Number Theorem, since for sufficiently large x we have $x/p > Cx/\ln x$. Thus the set of primes has arbitrarily large gaps, and since words in L' have prime length its gaps are at least as large. This completes the proof. □