CSE 547T Automata Theory March 2, 2017

Solutions for Homework 2

1. (a) Let n be given, and set $x = 0^n 10^{n+1}$. Let uvw be any valid decomposition of x; v must consist entirely of 0's. Observe that $z = uv^2w = 0^k 10^{n+1}$, where $k > n$; hence, $z \notin L$. Conclude that L is not regular.

   (b) Let n be given, and set $x = 0^n 110^n$. Let uvw be any valid decomposition of x; v must consist entirely of 0's. Observe that $z = uv^2w = 0^k 110^n$, where $k > n$; hence, the first half of z consists entirely of 0's, while the second half has two 1's. Conclude that $z \notin L$, and so L is not regular.

   (c) Let n be given, and let p be any prime number $\geq n + 2$. Set $x = 0^p 1^p$. Let uvw be any valid decomposition of x; v must consist entirely of 0's. Observe that $z = uv^0w = 0^k 1^p$, where $2 \leq k \leq p - 1$. Because p is prime, k cannot divide p; hence, $z \notin L$. Conclude that L is not regular.

2. **Claim 1**: the stated property implies that L_g is regular.

   Let m be the assumed value for which $g(n + k) = g(n)$ for all $n \geq m$. Note that there are at most $m + k - 1$ distinct values taken by the function g, since the values cycle with period k starting with m.

   Let M_i be a linear NFA of length $g(i) + 1$, $0 \leq i < m + k$, that has transitions only ones and accepts at its last state. More specifically, M_i consists of states $q_0 \ldots q_{g(i)}$, with $\delta(q_k, 1) = \{q_{k+1}\}$, and $q_{g(i)}$ accepts but has no outgoing transitions. M_i manifestly accepts precisely the string $1^{g(i)}$.

   Construct the following ε-NFA M to accept L_g. M consists of a base NFA M_b with exactly $m + k - 1$ states, attached to the machines M_i as follows. The states of M_b are organized in a linear chain of zero transitions with a single back edge from the last state to the nth state. The ith state in the chain has a ε transition to the start state of machine M_i.

   More precisely, M_b consists of states $q_0 \ldots q_{m+k-1}$, with $\delta(q_j, 0) = \{q_{j+1}\}$ for $j < m + k - 1$, and $\delta(q_{m+k-1}, 0) = \{q_m\}$. We also have that $\delta(q_j, \varepsilon)$ is the start state of M_i. None of the states in M_b accept, but the accepting states of each M_i are preserved as above.

   We claim that M accepts L_g. For any string x of the form $0^n 1^{g(n)}$, the following computation is accepting. Suppose that M takes no ε transitions (i.e. stays in M_b) until all zeros have been read. If $n < m$, M_b ends in state q_n; otherwise, it ends in state q_{m+j}, where $j = (n - m) \bmod k$. M then takes the lambda transition to the corresponding linear machine (M_n if $n < m$, or M_{m+j} if $n \geq m$). By construction, this machine accepts $1^{g(n)}$, which is the remainder of x.

   Conversely, if M accepts x, then x consists of some number of zeros followed by some number of ones, since there are no links back from the M_i's to M_b. The accepting path therefore moves to some state q_p of M_b and then takes the linear NFA M_p to its end. By construction, M_p accepts precisely $1^{g(p)}$. If $p < m$, then q_p is reachable only by a string 0^p, and so x has form $0^p 1^{g(p)}$ as desired. Otherwise, q_p is reachable by any string of the form 0^n, where $n = m + jk + (p - m)$. By our assumptions about g, $g(n) = g(p)$. Conclude that $x = 0^n 1^{g(n)}$, as desired.

   Claim 2: any regular language L_g of the specified form has the specified property.

   We use the proof of the Pumping Lemma as follows. Let M be a DFA accepting L_g, and suppose M has exactly m states. If $n \geq m$, the path taken by $x_n = 0^n 1^{g(n)}$ in M has at least m edges and so

contains a cycle within its zero-reading portion. Let k_n be the length of this cycle. Then $0^{n+jk_n}1^{g(n)}$ is also accepted by M, and so it must be that $g(n + jk_n) = g(n)$ for all $j > 0$.

However, the claim requires that we find a *single* k such that $g(n + k) = g(n)$ for *all* large enough n. Observe that, since each k_n is the length (in edges) of a cycle in M, it must be that $k_n \leq m$, and so there are only finitely many distinct values of k_n possible. Let $K = \prod_{j=2}^{m} j$; then each possible k_n surely divides K. Let n^* be the largest among the thresholds n associated with all distinct values of k_n. Then for *all* $n \geq n^*$, we have that $g(n + jK) = g(n)$ for all $j > 0$, which proves the claim.

3. (a) We claim that L has exactly the following equivalence classes:

$$\{[\varepsilon], [0], [00], [01]\}.$$

The following table gives, for each pair of representative strings x and y from the above classes, a string z that distinguishes them.

	0	00	01
ε	1	ε	ε
0		ε	ε
00			1

It remains to show that these classes are exhaustive, i.e. that every string $x \in \Sigma^*$ is in some class. First, suppose $|x| < 2$. ε and 0 are in the classes with those labels, by definition. We further claim that $1 \in [\varepsilon]$. Indeed, if $\varepsilon z \in L$, so too is $1z$; conversely, if $1z \in L$, then z must have at least two characters (since the second-to-last may not be 1), and so $z \in L$.

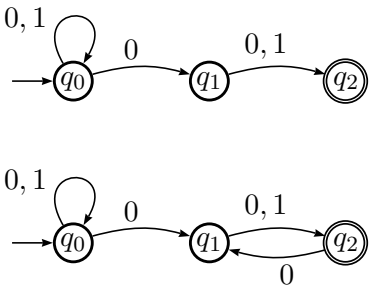
Now suppose $|x| \geq 2$. We can characterize the class of x by considering the fate of all xz for $|z| < 2$. If $|z| \geq 2$, then the fate of xz depends only on z, regardless of x.

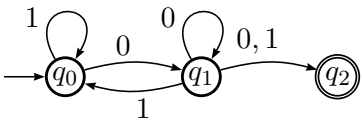
- If x ends in 01, then $x \in L$, but neither $x0$ nor $x1$ is in L. This behavior implies that $[x] = [01]$.
- if x ends in 11, then none of x, $x0$ or $x1$ are in L. This behavior implies that $[x] = [\varepsilon]$.
- if x ends in 00, then all of x, $x0$ or $x1$ are in L. This behavior implies that $[x] = [00]$.
- if x ends in 10, then $x \notin L$, but $x0$ and $x1$ are both in L. This behavior implies that $[x] = [0]$.

The resulting DFA is as per the Myhill-Nerode construction, with $q_0 = \langle[\varepsilon]\rangle$, $A = \{\langle[00]\rangle, \langle[01]\rangle\}$ and the following δ:

	0	1
$\langle[\varepsilon]\rangle$	$\langle[0]\rangle$	$\langle[\varepsilon]\rangle$
$\langle[0]\rangle$	$\langle[00]\rangle$	$\langle[01]\rangle$
$\langle[00]\rangle$	$\langle[00]\rangle$	$\langle[01]\rangle$
$\langle[01]\rangle$	$\langle[0]\rangle$	$\langle[\varepsilon]\rangle$

(b) Here are three alternative NFA designs with three states:





These NFAs are non-isomorphic, in the sense that we cannot find a 1:1 correspondence between their states that keeps the outgoing transitions the same for each corresponding state pair.

We first claim that three states is the minimal size of any NFA accepting L. Suppose there exists an NFA M for L with only two states. At least one (but not both) of the states must be accepting. We cannot start in the accepting state, since $\varepsilon \notin L$. But if there is any transition from the start state to the accepting state, then M accepts a string of length 1, which is not in L.

We now claim that all three NFAs shown accept L. For the top NFA (call it M_1), if $z = x0a \in L$, then we can accept z by consuming x in state q_0, then transitioning to q_1 and then q_2 on $0a$. Conversely, suppose M_1 accepts z. It must have at least two characters (which must be the last two of the string) to get from q_0 to q_2, and the first of these must be 0. Hence, $z \in L$.

For the middle NFA (call it M_2), the same proof as for M_1 shows that $z = x0a \in L$. For the converse, suppose M_2 accepts z. As before, z must have at least two characters to get from q_0 to q_2. After reading all but the last two characters, suppose M_2 is in one of its three states. If it accepts in two steps starting from q_0 *or* q_2, the first of the two characters consumed must be a 0. Moreover, there is no way to accept in exactly two steps starting from q_1. Hence, every string accepted by M_2 must have its second-to-last character be a 0.

For the bottom NFA (call it M_3), if $z = x0a \in L$, then the initial portion x of z leaves M_3 in q_0 (if x ends with 0) or q_1 (if x ends with 1 or is ε). In the former case, the final $0a$ is consumed by a self-loop followed by the transition to q_2. In the latter case, the final $0a$ is consumed by transitioning from q_0 to q_1 to q_2. Conversely, suppose M_3 accepts z. It must end with at least two characters to get from q_0 to q_2; hence, let $z = xba$. Observe that, regardless of whether $\delta^*(q_0, x)$ is q_0 or q_1, only two-character strings with $b = 0$ can move M to q_2. Hence, $z \in L$.

4. (a) Consider the strings 0^i1, for $i \geq 0$. The string 0^i1 is completed by $z = 0^i1$, but z does not complete 0^j1 for any $j \neq i$. Hence, every 0^i1 is in a different equivalence class, and so there are infinitely many such classes. Conclude that L is not regular.

   (b) Consider the strings 0^i1 for $i \geq 0$. Observe that the string $0^i1(10^j1)$ is in L for $j \geq i$, since it has the form 0^i1c0^i1, but is not in L for $j < i$.

   Setting $z = 10^j1$, we have that z distinguishes the string 0^j1 from every string 0^i1, $i > j$. Hence, if $i \neq j$, the strings 0^j1 and 0^i1 are distinguishable. Conclude that there are infinitely many equivalence classes of L, and so it is not regular.

   (c) This language is regular. We claim that its equivalence classes are precisely the following:

$$\{[\varepsilon], [0], [1], [01], [10], [000], [111]\}.$$

   The following table shows that the representative strings for our classes are all pairwise distinguished. For each pair of strings x and y, labeling a row and column of the table respectively, we give a distinguishing string z.

	0	1	01	10	000	111
ε	00	11	0	1	ε	ε
0		11	0	1	ε	ε
1			0	1	ε	ε
01				1	ε	ε
10					ε	ε
000						0

It remains to show that these classes are exhaustive, i.e. that every string of Σ^* is in some class.

- Every string of length 0 or 1 is in its own class.
- Among strings of length 2, 01 and 10 are class labels. Observe that 01 is completed (only) by strings ending in 0, as is 00; similarly, 10 is completed (only) by strings ending in 1, as is 11. Hence, $00 \in [01]$ and $11 \in [10]$.
- For any string x with $|x| \geq 3$, we have one of four cases.
  i. $x = 0y0$; in this case, x is completed by ε or by $z0$ for any $z \in \Sigma^*$. Hence $x \in [000]$.
  ii. $x = 1y1$; in this case, x is completed by ε or by $z1$ for any $z \in \Sigma^*$. Hence $x \in [111]$.
  iii. $x = 0y1$; in this case, x is completed by $z0$ for any $z \in \Sigma^*$. Hence $x \in [01]$.
  iv. $x = 1y0$; in this case, x is completed by $z1$ for any $z \in \Sigma^*$. Hence $x \in [10]$.

5. (a) Augmented regular expressions can describe non-regular languages. Consider the language $L = \{ww \mid w \in \Sigma^*\}$, which was proven nonregular above. An expression matching L would be

$$\$(0 \cup 1)^*\$\beta.$$

By construction, a string matches this expression iff its first and second halves are identical.

(b) We claim that, if there no Kleene closures allowed in delimited subexpressions, then augmented regular expressions are equivalent to ordinary regular expressions.

Suppose R is an ordinary regular expression without Kleene closure. Because R is built using only concatenation and finite union from a finite number of base cases, it matches only a finite number k of strings. Denote these strings by $r_1 \ldots r_k$.

To convert an arbitrary augmented regular expression T to an ordinary expression T', we proceed inductively as follows:

- If T is an ordinary expression, set $T' = T$.
- If $T = \$R\$(S)\beta$, inductively convert S to an ordinary expression (S'), then form the union expression T' as follows:

$$T' = r_1(S')r_1 \cup \ldots \cup r_k(S')r_k.$$

For every z matching T, we can divde $z = uvw$, such that u matches R, v matches S, and w matches the backreference (and hence $w = u$). Inductively, we have that v matches S'; moreover, since T' contains one case for every string matching R, it contains a case of the form $u(S')w$. Conversely, if z matches T', we can divide $z = r_j v r_j$, where v matches S'. Since r_j matches R, and S is equivalent to S', we have that z matches T as well.

- Finally, to convert the union, concatenation, or Kleene closure of augmented regular expressions, convert their component parts.