CS 341 Homework 9
Languages That Are and Are Not Regular

1. Show that the following are not regular.
(a) $L = \{ww^R : w \in \{a, b\}^*\}$
(b) $L = \{ww : w \in \{a, b\}^*\}$
(c) $L = \{ww' : w \in \{a, b\}^*\}$, where w' stands for w with each occurrence of a replaced by b, and vice versa.

2. Show that each of the following is or is not a regular language. The decimal notation for a number is the number written in the usual way, as a string over the alphabet {-, 0, 1, …, 9}. For example, the decimal notation for 13 is a string of length 2. In unary notation, only the symbol 1 is used; thus 5 would be represented as 11111 in unary notation.
(a) $L = \{w : w$ is the unary notation for a natural number that is a multiple of 7$\}$
(b) $L = \{w : w$ is the decimal notation for a natural number that is a multiple of 7$\}$
(c) $L = \{w : w$ is the unary notation for a natural number n such that there exists a pair p and q of twin primes, both > n.$\}$ Two numbers p and q are a pair of twin primes iff q = p + 2 and both p and q are prime. For example, (3, 5) is a pair of twin primes.
(d) $L = \{w : w$ is, for some $n \geq 1$, the unary notation for $10^n\}$
(e) $L = \{w : w$ is, for some $n \geq 1$, the decimal notation for $10^n\}$
(f) $L = \{w$ is of the form x#y, where x, $y \in \{1\}^+$ and y = x+1 when x and y are interpreted as unary numbers$\}$ (For example, 11#111 and 1111#11111 $\in$ L, while 11#11, 1#111, and 1111 $\notin$ L.)
(g) $L = \{a^n b^j : |n - j| = 2\}$
(h) $L = \{uww^Rv : u, v, w \in \{a, b\}^+\}$
(i) $L = \{w \in \{a, b\}^* :$ for each prefix x of w, $\#a(x) \geq \#b(x)\}$

3. Are the following statements true or false? Explain your answer in each case. (In each case, a fixed alphabet Σ is assumed.)
(a) Every subset of a regular language is regular.
(b) Let $L' = L1 \cap L2$. If L′ is regular and L2 is regular, L1 must be regular.
(c) If L is regular, then so is $L' = \{xy : x \in L$ and $y \notin L\}$.
(d) $\{w : w = w^R\}$ is regular.
(e) If L is a regular language, then so is $L' = \{w : w \in L$ and $w^R \in L\}$.
(f) If C is any set of regular languages, $\cup$C (the union of all the elements of C) is a regular language.
(g) $L = \{xyx^R : x, y \in \Sigma^*\}$ is regular.
(h) If $L' = L1 \cup L2$ is a regular language and L1 is a regular language, then L2 is a regular language.
(i) Every regular language has a regular proper subset.
(j) If L1 and L2 are nonregular languages, then $L1 \cup L2$ is also not regular.

4. Show that the language $L = \{a^n b^m : n \neq m\}$ is not regular.

5. Prove or disprove the following statement:
If L_1 and L_2 are not regular languages, then $L_1 \cup L_2$ is not regular.

6. Show that the language $L = \{x \in \{a, b\}^* : x = a^n b a^m b a^{max(m,n)}\}$ is not regular.

7. Show that the language $L = \{x \in \{a, b\}^* :$ x contains exactly two more b's than a's$\}$ is not regular.

8. Show that the language $L = \{x \in \{a, b\}^* :$ x contains twice as many a's as b's$\}$ is not regular.

9. Let L = {w : #a(w) = #b(w)}. (#a(w) = the number of a's in w.)
(a) Is L regular?
(b) Is L* regular?

Solutions

1. (a) L = $\{ww^R : w \in \{a, b\}^*\}$. L is the set of all strings whose first half is equal to the reverse of the second half. All strings in L must have even length. If L is regular, then the pumping lemma tells us that $\exists$ N $\geq$ 1, such that $\forall$ strings w $\in$ L, where |w| $\geq$ N, $\exists$ x, y, z, such that w = xyz, |xy| $\leq$ N, y $\neq \varepsilon$, and $\forall$ q $\geq$ 0, xy^qz is in L. We must pick a string w $\in$ L and show that it does not meet these requirements.

First, don't get confused by the fact that we must pick a string w, yet we are looking for strings of the form ww^R. These are two independent uses of the variable name w. It just happens that the problem statement uses the same variable name that the pumping lemma does. If it helps, restate the problem as L = $\{ss^R : s \in \{a, b\}^*\}$.

We need to choose a "long" w, i.e., one whose length is greater than N. But it may be easier if we choose one that is even longer than that. Remember that the fact that |xy| $\leq$ N guarantees that y (the pumpable region) must occur within the first N characters of w. If we don't want to have to consider a lot of different possibilities for what y could be, it will help to choose a w with a long first region. Let's let w = a^Nbba^N. We know that y must consist of one or more a's in the region before the b's. Clearly if we pump in any extra a's, we will no longer have a string in L. Thus we know that L is not regular.

Notice that we could have skipped the b's altogether and chosen w = a^Na^N. Again, we'd know that y must be a string of one or more a's. Unfortunately, if y is of even length (and it could be: remember we don't get to pick y), then we can pump in all the copies of y we want and still have a string in L. Sure, the boundary between the first half and the second half will move, that that doesn't matter. It is usually good to choose a string with a long, uniform first region followed by a definitive boundary between it and succeeding regions so that when you pump, it's clearly the first region that has changed.

(b) L = $\{ww : w \in \{a, b\}^*\}$. We'll use the pumping lemma. Again, don't get confused by the use of the variable w both to define L and as the name for the string we will choose to pump on. As is always the case, the only real work we have to do is to choose an appropriate string w. We need one that is long enough (i.e., |w| $\geq$ N). And we need one with firm boundaries between regions. So let's choose w = a^Nba^Nb. Since |xy| $\leq$ N, we know that y must occur in the first a region. Clearly if we pump in any additional a's, the two halves of w will no longer be equal. Q. E. D. By the way, we could have chosen other strings for w. For example, let w = ba^Nba^N. But then there are additional choices for what y could be (since y could include the initial b) and we would have to work through them all.

(c) L = $\{ww' : w \in \{a, b\}^*\}$, where w' stands for w with each occurrence of a replaced by b, and vice versa. We can prove this easily using the pumping lemma. Let w = a^Nb^N. Since |xy| $\leq$ N, y must be a string of all a's. So, when we pump (either in or out), we modify the first part of w but not the second part. Thus the resulting string is not in L.

We could also solve this problem just by observing that, if L is regular, so is L′ = L $\cap$ a*b*. But L′ is just a^nb^n, which we have already shown is not regular. Thus L is not regular either.

2. (a) L = {w : w is the unary notation for a natural number that is a multiple of 7}. L is regular since it can be described by the regular expression (1111111)*.

(b) L = {w : w is the decimal notation for a natural number that is a multiple of 7}. L is regular. We can build a deterministic FSM M to accept it. M is based on the standard algorithm for long division. The states represent the remainders we have seen so far (so there are 7 of them, corresponding to 0 – 6). The start state, of course, is 0, corresponding to a remainder of 0. So is the final state. The transitions of M are as follows:

$\forall s_i \in \{0 - 6\}$ and $\forall c_j \in \{0 - 9\}$, $\delta(s_i, c_j) = (10s_i + c_j) \bmod 7$

So, for example, on the input 962, M would first read 9. When you divide 7 into 9 you get 1 (which we don't care about since we don't actually care about the answer – we just care whether the remainder is 0) with a remainder of 2. So M will enter state 2. Next it reads 6. Since it is in state 2, it must divide 7 into 2*10 +6 (26). It gets a remainder of 5, so it goes to state 5. Next it reads 2. Since it is in state 5, it must divide 7 into 5*10 + 5 (52), producing a remainder of 3. Since 3 is not zero, we know that 862 is not divisible by 7, so M rejects.

(c) L = {w : w is the unary notation for a natural number such that there exists a pair p and q of twin primes, both > n.}. L is regular. Unfortunately, this time we don't know how to build a PDA for it. We can, however, prove that it is regular by considering the following two possibilities:

(1) There is an infinite number of twin primes. In this case, for every n, there exists a pair of twin primes greater than n. Thus L = 1*, which is clearly regular.
(2) There is not an infinite number of twin primes. In this case, there is some largest pair. There is thus also a largest n that has a pair greater than it. Thus the set of such n's is finite and so is L (the unary encodings of those values of n). Since L is finite, it is clearly regular.

It is not known which of these cases is true. But interestingly, from our point of view, it doesn't matter. L is regular in either case. It may bother you that we can assert that L is regular when we cannot draw either an FSM or a regular expression for it. It shouldn't bother you. We have just given a nonconstructive proof that L is regular (and thus, by the way, that some FSM M accepts it). Not all proofs need to be constructive. This situation isn't really any different from the case of L′ = {w : w is the unary encoding of the number of siblings I have}. You know that L′ is finite and thus regular, even though you do not know how many siblings I have and thus cannot actually build a machine to accept L′.

(d) L = {w : w is, for some n ≥ 1, the unary notation for 10^n}. So L = {1111111111, 1^{100}, 1^{1000}, …}. L isn't regular, since clearly any machine to accept L will have to count the 1's. We can prove this using the pumping lemma: Let $w = 1^P$, N ≤ P and P is some power of 10. y must be some number of 1's. Clearly, it can be of length at most P. When we pump it in once, we get a string s whose maximum length is therefore 2P. But the next power of 10 is 10P. Thus s cannot be in L.

(e) L = {w : w is, for some n ≥ 1, the decimal notation for 10^n}. Often it's easier to work with unary representations, but not in this case. This L is regular, since it is just 100*.

(f) L = {w is of the form x#y, where x, y ∈ $\{1\}^+$ and y = x+1 when x and y are interpreted as unary numbers} (For example, 11#111 and 1111#11111 ∈ L, while 11#11, 1#111, and 1111 ∉ L.) L isn't regular. Intuitively, it isn't regular because any machine to accept it must count the 1's before the # and then compare that number to the number of 1's after the #. We can prove that this is true using the pumping lemma: Let $w = 1^N\#1^{N+1}$. Since |xy| ≤ N, y must occur in the region before the #. Thus when we pump (either in or out) we will change x but not make the corresponding change to y, so y will no longer equal x +1. The resulting string is thus not in L.

(g) L = $\{a^n b^j : |n - j| = 2\}$. L isn't regular. L consists of all strings of the form a*b* where either the number of a's is two more than the number of b's or the number of b's is two more than the number of a's. We can show that L is not regular by pumping. Let $w = a^N b^{N+2}$. Since |xy| ≤ N, y must equal a^p for some p > 0. We can pump y out once, which will generate the string $a^{N-p}b^{N+2}$, which is not in L.

(h) L = {uww^{R}v : u, v, w ∈ {a, b}+}. L is regular. This may seem counterintuitive. But any string of length at least four with two consecutive symbols, not including the first and the last ones, is in L. We simply make everything up to the first of the two consecutive symbols u. The first of the two consecutive symbols is w. The second is w^R. And the rest of the string is v. And only strings with at least one pair of consecutive symbols (not including the first and last) are in L because w must end with some symbol s. w^R must start with that same symbol s. Thus the string will contain two consecutive occurrences of s. L is regular because it can be described the regular expression $(a \cup b)^+ (aa \cup bb) (a \cup b)^+$.

(i) L = {w ∈ {a, b}* : for each prefix x of w, #a(x) ≥ #b(x)}. First we need to understand exactly what L is. In order to do that, we need to define prefix. A string x is a prefix of a string y iff ∃z ∈ Σ* such that y = xz. In other words, x is a prefix of y iff x is an initial substring of y. For example, the prefixes of abba are ε, a, ab, abb, and abba. So L is all strings over {a, b}* such that, at any point in the string (reading left to right), there have never been more b's than a's. The strings ε, a, ab, aaabbb, and ababa are in L. The strings b, ba, abba, and ababb are not in L. L is not regular, which we can show by pumping. Let w = $a^N b^N$. So y = a^p, for some nonzero p. If we pump out, there will be fewer a's than b's in the resulting string s. So s is not in L since every string is a prefix of itself.

3. (a) Every subset of a regular language is regular. FALSE. Often the easiest way to show that a universally quantified statement such as this is false by showing a counterexample. So consider L = a*. L is clearly regular, since we have just shown a regular expression for it. Now consider L′ = a^i: i is prime. L′ ⊆ L. But we showed in class that L′ is not regular.

(b) Let L′ = L1 ∩ L2. If L′ is regular and L2 is regular, L1 must be regular. FALSE. We know that the regular languages are closed under intersection. But it is important to keep in mind that this closure lemma (as well as all the others we will prove) only says exactly what it says and no more. In particular, it says that:

If L1 is regular and L2 is regular
Then L′ is regular.

Just like any implication, we can't run this one backward and conclude anything from the fact that L′ is regular. Of course, we can't use the closure lemma to say that L1 must not be regular either. So we can't apply the closure lemma here at all. A rule of thumb: it is almost never true that you can prove the converse of a closure lemma. So it makes sense to look first for a counterexample. We don't have to look far. Let L′ = ∅. Let L2 = ∅. So L′ and L2 are regular. Now let L1 = {a^i: i is prime}. L1 is not regular. Yet L′ = L1 ∩ L2. Notice that we could have made L2 anything at all and its intersection with ∅ would have been ∅. When you are looking for counterexamples, it usually works to look for very simple ones such as ∅ or Σ*, so it's a good idea to start there first. ∅ works well in this case because we're doing intersection. Σ* is often useful when we're doing union.

(c) If L is regular, then so is L′ = {xy : x ∈ L and y ∉ L}. TRUE. Proof: Saying that y ∉ L is equivalent to saying that y ∈ $\overline{L}$. Since the regular languages are closed under complement, we know that $\overline{L}$ is also regular. L′ is thus the concatenation of two regular languages. The regular languages are closed under concatenation. Thus L′ must be regular.

(d) L = {w : w = w^R} is regular. FALSE. L is NOT regular. You can prove this easily by using the pumping lemma and letting w = $a^N b a^N$.

(e) If L is a regular language, then so is L′ = {w : w ∈ L and w^R ∈ L}. TRUE. Proof: Saying that w^R ∈ L is equivalent to saying that w ∈ L^R. If w must be in both L and L^R, that is equivalent to saying that L′ = L ∩ L^R. L is regular because the problem statement says so. L^R is also regular because the regular languages are closed

under reversal. The regular languages are closed under intersection. So the intersection of L and L^R must be regular.

Proof that the regular languages are closed under reversal (by construction): If L is regular, then there exists some FSM M that accepts it. From M, we can construct a new FSM M′ that accepts L^R. M′ will effectively run M backwards. Start with the states of M′ equal to states of M. Take the state that corresponds to the start state of M and make it the final state of M′. Next we want to take the final states of M and make them the start states of M′. But M′ can have only a single start state. So create a new start state in M′ and create an epsilon transition from it to each of the states in M′ that correspond to final states of M. Now just flip the arrows on all the transitions of M and add these new transitions to M′.

(f) If C is any set of regular languages, ∪C is a regular language. FALSE. If C is a *finite* set of regular languages, this is true. It follows from the fact that the regular languages are closed under union. But suppose that C is an infinite set of languages. Then this statement cannot be true. If it were, then every language would be regular and we have proved that there are languages that are not regular. Why is this? Because every language is the union of some set of regular languages. Let L be an arbitrary language whose elements are w_1, w_2, w_3, …. Let C be the set of singleton languages $\{\{w_1\}, \{w_2\}, \{w_3\}, \dots\}$ such that $w_i \in L$. The number of elements of C is equal to the cardinality of L. Each individual element of C is a language that contains a single string, and so it is finite and thus regular. L = ∪C. Thus, since not all languages are regular, it must not be the case that ∪C is guaranteed to be regular. If you're not sure you follow this argument, you should try to come up with a specific counterexample. Choose an L such that L is not regular, and show that it can be described as ∪C for some set of languages C.

(g) $L = \{xyx^R : x, y \in \Sigma^*\}$ is regular. TRUE. Why? We've already said that xx^R isn't regular. This looks a lot like that, but it differs in a key way. L is the set of strings that can be described as some string x, followed by some string y (where x and y can be chosen completely independently), followed by the reverse of x. So, for example, it is clear that abccccba ∈ L (assuming Σ ={a, b, c}). We let x = ab, y = ccccc, and x^R = ba. Now consider abbccccaaa. You might think that this string is not in L. But it is. We let x = a, y = bbccccaa, and x^R = a. What about acccb? This string too is in L. We let x = ε, y = acccb, and x^R = ε. Note the following things about our definition of L: (1) There is no restriction on the length of x. Thus we can let x = ε. (2)There is no restriction on the relationship of y to x. And (3) $\varepsilon^R = \varepsilon$. Thus L is in fact equal to Σ* because we can take any string w in Σ* and rewrite it as ε w ε, which is of the form xyx^R. Since Σ* is regular, L must be regular.

(h) If L′ = L1 ∪ L2 is a regular language and L1 is a regular language, then L2 is a regular language. FALSE. This is another attempt to use a closure theorem backwards. Let L1 = Σ*. L1 is clearly regular. Since L1 contains all strings over Σ, the union of L1 with any language is just L1 (i.e., L′ = Σ*). If the proposition were true, then all languages L2 would necessarily be regular. But we have already shown that there are languages that are not regular. Thus the proposition must be false.

(i) Every regular language has a regular proper subset. FALSE. ∅ is regular. And it is subset of every set. Thus it is a subset of every regular language. However, it is not a *proper* subset of itself. Thus this statement is false. However the following two similar statements are true:

(1) Every regular language has a regular subset.
(2) Every regular language except ∅ has a regular proper subset.

(j) If L1 and L2 are nonregular languages, then L1 ∪ L2 is also not regular. False. Let $L1 = \{a^n b^m, n \geq m\}$ and $L2 = \{a^n b^m, n \leq m\}$. L1 ∪ L2 = a*b*, which is regular.

4. If L were regular, then its complement, L_1, would also be regular. L_1 contains all strings over {a, b} that are not in L. There are two ways not to be in L: have any a's that occur after any b's (in other words, not have all the a's followed by all the b's), or have an equal number of a's and b's. So now consider

$L_2 = L_1 \cap a^*b^*$

L_2 contains only those elements of L_1 in which the a's and b's are in the right order. In other words,

$L_2 = a^nb^n$

But if L were regular, then L_1 would be regular. Then L_2, since it is the intersection of two regular languages would also be regular. But we have already shown that it (a^nb^n) is not regular. Thus L cannot be regular.

5. This statement is false. To prove it, we offer a counter example. Let $L_1 = \{a^nb^m : n=m\}$ and let $L_2 = \{a^nb^m : n \neq m\}$. We have shown that both L_1 and L_2 are not regular. However,

$L_1 \cup L_2 = a^*b^*$, which is regular.

There are plenty of other examples as well. Let $L_1 = \{a^n: n \geq 1$ and n is prime$\}$. Let $L_2 = \{a^n: n \geq 1$ and n is not prime$\}$. Neither L_1 nor L_2 is regular. But $L_1 \cup L_2 = a^+$, which is clearly regular.

6. This is easy to prove using the pumping lemma. Let $w = a^Nba^Nba^N$. We know that xy must be contained within the first block of a's. So, no matter how y is chosen (as long as it is not empty, as required by the lemma), for any $i > 2$, $xy^iz \notin L$, since the first block of a's will be longer than the last block, which is not allowed. Therefore L is not regular.

7. First, let $L' = L \cap a^*b^*$, which must be regular if L is. We observe that $L' = a^nb^{n+2} : n \geq 0$. Now use the pumping lemma to show that L' is not regular in the same way we used it to show that a^nb^n is not regular.

8. We use the pumping lemma. Let $w = a^{2N}b^N$. xy must be contained within the block of a's, so when we pump either in or out, it will no longer be true that there will be twice as many a's as b's, since the number of a's changes but not the number of b's. Thus the pumped string will not be in L. Therefore L is not regular.

9. **(a)** L is not regular. We can prove this using the pumping lemma. Let $w = a^Nb^N$. Since y must occur within the first N characters of w, $y = a^p$ for some $p > 0$. Thus when we pump y in, we will have more a's than b's, which produces strings that are not in L.

(b) L* is also not regular. To prove this, we need first to prove a lemma, which we'll call EQAB: $\forall s, s \in L^* \Rightarrow \#a(s) = \#b(s)$. To prove the lemma, we first observe that any string s in L* must be able to be decomposed into at least one finite sequence of strings, each element of which is in L. Some strings will have multiple such decompositions. In other words, there may be more than one way to form s by concatenating together strings in L. For any string s in L*, let SQ be some sequence of elements of L that, when concatenated together, form s. It doesn't matter which one. Define the function HowMany on the elements of L*. HowMany(x) returns the length of SQ. Think of HowMany as telling you how many times we went through the Kleene star loop in deriving x. We will prove EQAB by induction on HowMany(s).

Base case: If HowMany(s) = 0, then $s = \varepsilon$. #a(s) = #b(s).

Induction hypothesis: If HowMany(s) $\leq$ N, then #a(s) = #b(s).

Show: If HowMany(s) = N+1, then #a(s) = #b(s).

If HowMany(s) = N+1, then $\exists w,y$ such that $s = wy$, $w \in L^*$, HowMany(w) = N, and $y \in L$. In other words, we can decompose s into a part that was formed by concatenating together N instances of L plus a second part that is just one more instance of L. Thus we have:

(1) #a(y) = #b(y).	Definition of L	
(2) #a(w) = #b(w).	Induction hypothesis	
(3) #a(s) = #a(w) + #a(y)	s = wy	
(4) #b(s) = #b(w) + #b(y).	s = wy	
(5) #b(s) = #a(w) + #b(y)	4, 2	
(6) #b(s) = #a(w) + #a(y)	5, 1	
(7) #b(s) = #a(s)	6, 3	Q. E. D.

Now we can show that L* isn't regular using the pumping lemma. Let $w = a^N b^N$. Since y must occur within the first N characters of w, $y = a^p$ for some $p > 0$. Thus when we pump y in, we will have a string with more a's than b's. By EQAB, that string cannot be in L*.