Main topics of the week:

- **Pumping lemma and examples**
- **State minimization**
- **Examples of proving regularity**

Pumping Lemma Examples

Consider the language $B = \{0^n1^n \mid n \geq 0\}$ that we intuitively feel is not regular. Suppose that B is regular. Then by the pumping lemma, there is a pumping length p. Consider the string 0^p1^p, and let x, y, and z be the substrings guaranteed by the pumping lemma. It is certainly true that y must consist of just zeroes, or just ones, or both. If y consists of just zeroes, then the string xyyz will have have more zeroes than ones since the y adds only more zeroes, and x and z are fixed. Likewise, if y consists of just ones, then the same reasoning shows that xyyz has more ones than zeroes. So we know that y cannot be all zeroes or all ones. So y must have some of each. However, if we again look at the string xyyz, we would see that the zeroes of the second y occurrence would occur after the ones in the first y occurrence, again a contradiction. Since all of the possibilities for y lead to a contradiction, we conclude that there can be no pumping length, hence B is not a regular language.

This proof can be simplified by using condition 3 from the pumping lemma: that $|xy| \leq p$. Using this, we can assert that y must consist entirely of zeroes and thus not have to argue all the separate cases.

The art of using the pumping lemma is being clever about the choice of the string that produces a contradiction. Above it was pretty easy since any string of length p would have worked, but that is not always the case.

For another example, let P be the language of all palindromes, i.e., strings that are the same when read in the reverse direction. If P is regular, then we have a pumping length p. Consider the string a^pba^p. Certainly this is a palindrome. In our decomposition xyz, we are guaranteed that the length of xy does not exceed p. Thus, xy must consist just of a's and in particular this will be true of y, say $y=a^k$, where $k \geq 1$. But then xz would have the form $a^{p-k}ba^p$, and this is obviously not a palindrome, so P is not regular.

Here's another example that uses a little more reasoning about lengths. Let $T = \{0^k \mid k = 2^n \text{ for some } n \geq 0\}$. That is, T is the set of all strings of zeroes whose length is a power of 2. If it is regular, then the pumping lemma applies. Let s be a string of length $m \geq p$, and s=xyz. We know that $y = 0^n$ for some n and $m = 2^k$ for some $k \geq 0$. Note that xz will have length $2^k - n$, which must also be a power of 2, say $2^k - n = 2^i$ for some $i < k$. Similarly, if we look at xyyz, we see that $2^k + n = 2^j$ for some $j > i$. Adding these together gives $2^{k+1} = 2^i + 2^j = 2^i (1 + 2^{j-i})$. Dividing out the 2^i leaves us with the left hand side a power of two, hence even, and the right hand side as 1 plus a power of two (power at least one), which is odd. Thus T is not regular.

State Minimization

The following material is based on problems 1.34 and 1.35 in Sipser.

Definition: Let x and y be any strings and L be any language. We say that x and y are **distinguishable** by L if some string z exists such that exactly one of the strings xz and yz is in L. Otherwise for every string z, $yz \in L \Rightarrow xz \in L$ and we say that x and y are **indistinguishable** by L. In this case, we write $x \equiv_L y$.

Lemma: Indistinguishability is an equivalence relation.

Proof: This is pretty obvious. Certainly it is reflexive and symmetric. And it's clearly transitive since if $x \equiv_L y$ and $y \equiv_L w$, and z is any string, and $wz \in L$, then $yz \in L$ by the indistinguishability of y and w, and thus $xz \in L$ by the indistinguishability of x and z. So $x \equiv_L w$.

Definition: Let L be a language and X a set of strings. We say that X is **pairwise distinguishable by L** if every two distinct strings in X are distinguishable by L. We define the **index of L** to be the maximum number of elements in any set that is pairwise distinguishable by L, and this could be finite or infinite.

Lemma: If L is recognized by a DFA with k states, then L has index at most k.

Proof: Let M be a k-state DFA that recognizes L. Suppose that L has index greater than k, i.e., there is a set X with more than k elements and X is pairwise distinguishable by L. Then there must be two distinct strings x and y in X which transition to the same state in M (by the pigeonhole principle, since there are only k states, but k+1 or more strings, they can't all transition to unique states). This would also imply that for any string z, xz and yz transition to the same state. In particular, if yz transitions to an accept state, so does xz, thus $x \equiv_L y$. But this contradicts the assumption of X being pairwise distinguishable by L. So our lemma is proved.

Lemma: If the index of L is a finite number k, then it is recognized by a DFA with k states.

Proof: We will construct a DFA M to recognize L. Let $X = \{s_1, s_2, \ldots, s_k\}$ be pairwise distinguishable by L (as guaranteed by the index of L being k). Define M as follows:

1) The states of M are $\{q_1, q_2, \ldots, q_k\}$ (i.e., one state for each string in X)
2) $\delta(q_i, a) = q_j$ where $s_j \equiv_L s_i a$. We know that s_j exists since otherwise $s_i a$ would be distinguishable from every element of X, meaning that $X \cup \{s_i a\}$ is a larger set pairwise distinguishable by L.
3) The accept states of M are $\{q_i \mid s_i \in L\}$
4) The start state of M is the q_i such that $s_i \equiv_L \varepsilon$.

We claim that M recognizes L. The start state corresponds to a string indistinguishable from ε. Each transition moves to the state corresponding to a string indistinguishable by appending the input character. And the accept states correspond to strings indistinguishable from the strings in L. Moreover, since $x \equiv_L s_i$ implies $xa \equiv_L s_i a$ for any a

($s_iaz = s_i(az) \in L \Rightarrow x(az) = xaz \in L$), we can see that the computation sequence for any string $a_1a_2\ldots a_n \in L$ is just the set of states associated with the equivalence classes of ε, a_1, a_1a_2, $a_1a_2a_3$, …, $a_1a_2\ldots a_n$ and the last corresponds to an accept state of M. For the other direction, any string accepted is likewise indistinguishable from the string of its acceptance state, namely a string in L, so must itself be in L (using ε in the indistinguishability definition).

Theorem: L is regular if and only if it has finite index. Moreover, its index is the size of the smallest DFA recognizing it.

Proof: By the first lemma, if L is regular, it has index at most k, where k is the number of states in a DFA recognizing it. By the second lemma, if L has finite index k, it is recognized by a DFA with k states, so is regular. As for the smallest part, suppose L has index k. Then we have a DFA with k states recognizing L by the construction in the lemma. If there was a DFA with a smaller number of states, then the first lemma would mean that the index of L was no more than that number of states, i.e., smaller than k. But we chose k to be the index of L, so this would be a contradiction.

So what do these results say? They suggest a way to find the minimal state DFA for a language by basing it on the equivalence classes of the indistinguishability relation. In practice, an algorithm to minimize the states of a DFA might proceed toward indistinguishability by degrees, i.e., indistinguishability by strings of length n, and using this to lump states together. Eventually, this algorithm would achieve a steady state, at which point the DFA would have minimal states.

Examples of proving regularity

Problem: Let Prefix(A) = { w | wx $\in$ A for some string x}, i.e., the initial parts of all strings in A. If A is a regular language, prove that Prefix(A) is a regular language.

Proof: Let M be a DFA that recognizes A. We construct the DFA M' that has the same states as M, the same transitions as M, and the same start state as M. The difference is in the accept states – the accept states of M' are all those states for which there is a path to an accept state of M. Note that this is well defined in that we can determine if a state is an accept state of M' by checking a finite number of transitions. To see that M' recognizes Prefix(A), let w $\in$ Prefix(A). Then wx $\in$ A, so there is a computation sequence in M for wx. The transitions for the x part lead to an accept state of M, hence the state beginning the x part is an accept state of M' by our definition. Thus the first part of the computation sequence shows w is accepted by M'. Conversely, if w is accepted by M', then there is a computation sequence resulting in an accept state of M'. By definition of the accept states of M', this means there is a transition sequence to an accept state of M, and putting these together gives the computation sequence of M, and x is the string associated with the latter part of the transition sequence, thus w $\in$ Prefix(A).

Problem: Prove that if A is a regular language, then the "proper" prefixes of A is a regular language, i.e., those prefixes that are not in A themselves.

Proof: Since regular languages are closed under complement, ~A is regular. And since regular languages are closed under intersection, ~A $\cap$ Prefix(A) is regular, and this is exactly the prefixes that are not in A.

Problem: Let A = { *w* | *w* contains an equal number of occurrences of the substrings **01** and **10**}. Note that **101** $\in$ A since it has a single **01** and a single **10**, but **1010** $\notin$ A since it has two **10**s, but only one **01**. Show that A is a regular language.

Proof: When you think about counting occurrences of 01 and 10 in strings, the presence of consecutive 1s and 0s does not affect the counting. That is, 0111100001 is equivalent to 0101 for the purposes of counting these substrings. Looking at it this way, we see that we are reducing to alternating 0s and 1s, and for the counts to be the same, the string must begin and end with the same symbol. So we really have the language described as beginning and ending with the same symbol, which is just the regular expression $0\Sigma^*0 \cup 1\Sigma^*1$, hence this is a regular language.