Main topics of the week:

- **Review problems for midterm**

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.

Problem: Let A be a regular language over the alphabet Σ. Show that B = { x | $x = yz$ for some $y \in$ A and some string $z \in \Sigma^*$} is a regular language.

Proof: Let R be a regular expression that describes A. Then $R\Sigma^*$ is a regular expression describing B. Alternatively, given an NFA for A, construct an NFA for B by adding a new accept state with ε transitions from the original accept states, and a loop for all symbols in Σ.

Problem: Let A be the language $\{a^i b^j a^k \mid k > i + j\}$. Prove that A is not regular.

Proof: Suppose that A is regular and let p be the value guaranteed by the pumping lemma. Consider the string $a^p b a^{p+2} \in$ A. When realized as xyz, where $|xy| \le p$ and $|y| > 0$, we see that y must be of the form a^k for some $k > 0$. Then xyyz would look like $a^{p+k} b a^{p+2}$, and since $k \ge 1$, p+k+1 cannot be strictly less than p+2, meaning that xyyz is not in the language, so A cannot be regular.

Problem: Let A be the language $\{ (ab)^n a^k \mid n > k \text{ and } k \ge 0 \}$. Prove that A is not regular.

Proof: Suppose that A is regular and let p be the pumping length. Consider the string $(ab)^{p+1} a^p \in$ A. Take any decomposition as xyz with $|xy| \le p$ and $|y| > 0$. Then we know that z ends with $(ab)a^p$. If y contains a b, then xz will have at most p b's, so cannot be in A. Likewise, if y contains an a, then xz will have at most p a's before the last b, so again cannot be in the language. Thus A cannot be regular.

Problem: Prove that the intersection of two regular languages is a regular language.

Proof: Let $M_{A=}(Q_A, \Sigma, \delta_A, q_A, F_A)$ and $M_{B=}(Q_B, \Sigma, \delta_B, q_B, F_B)$ be DFAs recognizing two regular languages A and B. Construct the DFA $N = (Q_B \times Q_B, \Sigma, \delta, (q_A,q_B), F_A \times F_B)$ where $\delta((p,q), a) = (\delta_A(p,a), \delta_B(q,b))$. If a string x is in $A \cap B$, then there is a computation sequence in M_A and also one in M_B. These two computation sequences describe a computation sequence in N leading to an accept state in N comprised of accept states from M_A and M_B. Conversely, an accepting computation sequence in N describes accepting computation sequences in both M_A and M_B. This is just like the construction of DFAs to show the union of regular languages is regular except that the accept states in the construction here require both elements of the pair of states to be accepting states for their respective machines. Alternatively, $A \cap B = \sim\sim(A \cap B) = \sim(\sim A \cup \sim B)$, a regular language since the complement of a regular language is regular as well as the union of two regular languages.

Problem: An All-Paths-NFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ like an NFA except that this automaton accepts a string if *every possible computation* of the string ends in an accept state. (Recall that in a regular NFA, if *some* computation ends in an accept state, then the string is accepted.) Prove that a language is regular if and only if it is recognized by an All-Paths-NFA.

Proof: If a language is regular, it is recognized by a DFA. Clearly, any DFA is also an All-Paths-NFA since in a DFA, a string has a unique computation sequence. For the other direction, given an All-Paths-NFA we will construct an equivalent DFA, using almost the same construction used to find an equivalent DFA for an NFA. Let $P = (Q, \Sigma, \delta, q_0, F)$ be the All-Paths-NFA. Then the constructed DFA $M = (2^Q, \Sigma, \delta', E(\{q_0\}), 2^F)$ recognizes the same language. This is just like the construction used to find an equivalent DFA for a given NFA except for the way the accept states are defined. The states of M are the subsets of Q, the start state is the ε closure of the start state of P, and the accept states of M are all the subsets of F. The transition function δ' is defined as $\delta'(R, a) = \{ q \mid q \in E(\delta(r, a)) \text{ for some } r \in R\}$. If a string s is accepted by P, then all paths through P result in acceptance. Thus s will be accepted by M since M was constructed to follow the paths of P, and so all of these paths resulting in accept states of P does constitute an accept state of M. Conversely, since the accept states of M consist only of accept states of P, all paths in P for an M-accepted string are accepting paths in P. Alternatively, we could define the NFA N to just be P with the accept and non-accept states reversed. Then because of the all paths property, N recognizes the complement of the language recognized by P. But since the complement of a regular language is regular, this means the language recognized by P is regular.