Main topics of the week:
- Review problems for midterm

**Problem**: Let $\Sigma = \{0, 1\}$ and $A = \{ w \in \Sigma^* | w$ does not contain a pair of 1’s separated by an odd number of symbols\}. Show that $A$ is a regular language.

**Proof**: Consider the complement of the language $A$, i.e., all strings that do contain a pair of 1’s separated by an odd number of symbols. We can write a regular expression for this complement as $(0 \cup 1)^* 1 (0 \cup 1) (0 \cup 1)^* 1 (0 \cup 1)^*$, showing the complement is a regular language. Since the complement of $A$ is regular, $A$ must be regular.

**Problem**: Let $A$ be a regular language over the alphabet $\Sigma$. 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 $\epsilon$ transitions from the original accept states, and a loop for all symbols in $\Sigma$.

**Problem**: Let $A$ be the language $\{ a^i b^j a^k | k > i + j \}$. 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 $|xy| \leq p$ and $|y| > 0$, we see that $y$ must be of the form $a^k$ for some $k > 0$. Then $xyz$ would look like $a^{p+k} b a^{p+2}$, and since $k \geq 1$, $p+k+1$ cannot be strictly less than $p+2$, meaning that $xyzz$ is not in the language, so $A$ cannot be regular.

**Problem**: Let $A$ be the language $\{ (ab)^n a^k | n > k$ and $k \geq 0 \}$. 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 $|xy| \leq p$ 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 reverse of a regular language is a regular language.

**Proof**: Let $A$ be a regular language and let $N = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing $A$. Define the NFA $N'$ to be $\langle Q' \cup \{ s \}, \Sigma, \delta', s, \{ q_0 \} \rangle$ where we define $\delta'$ by
\[
\delta'(q, a) = \begin{cases} 
    \{ p | \delta(p, a) = q \} & \text{if } q \neq s \\
    F & \text{if } q = s \text{ and } a = \epsilon \\
    \emptyset & \text{otherwise}
\end{cases}
\]
Basically, we are defining $N'$ to run $N$ in reverse. We add a new start state and $\epsilon$ transitions to the accept states of $N$, then reverse all the transitions of $N$. If a string is in
A, then its computation sequence ends in an accept state. When we feed the reverse of this string into $N'$, we choose the $\epsilon$ transition to the accept state of $N$, then choose the transitions corresponding to the ones from $N$, and these will lead us to the start state of $N$ which is the accept state of $N'$. Likewise, a string accepted by $N'$ describes a corresponding computation sequence in $N$.

**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 $M = (\mathcal{P}(Q), \Sigma, \delta', E(\{q_0\}), \mathcal{P}(F))$ 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 $\epsilon$ closure of the start state of $P$, and the accept states of $M$ are all the subsets of $F$. The transition function $\delta'$ 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.