Homework 1

Problem 0.5

For a set \( S \), the power set of \( S \) denoted \( \mathcal{P}(S) \) contains \( 2^{|S|} \) elements. The most straightforward technique is to show that a bijection exists between bit strings of length \( |S| \) and the elements of \( \mathcal{P}(S) \).

That is, we need to show that for any set \( S \)

- For any bit string of length \( |S| \) we can construct a unique subset of \( S \)
- For any subset of \( S \) we can construct a unique bit string of length \( |S| \)

To see this, assign an (arbitrary) order to the elements of \( S = \{s_1, \ldots, s_{|S|}\} \).
Now, consider any bit string of length \( |S| \), \( b = b_1 \ldots b_{|S|} \). Construct set \( A \) as follows:

\[
\forall i \quad s_i \in A \iff b_i = 1
\]

Note that \( A \subseteq S \) by construction.

Now, note that any subset \( A \) of \( S \) can be used to define a bit string \( b = b_1 \ldots b_{|S|} \) where

\[
b_i = \begin{cases} 
0 & \text{if } s_i \in A \\
1 & \text{otherwise}
\end{cases}
\]

Thus we have defined a bijection between bit strings of length \( |S| \) and the elements of \( \mathcal{P}(S) \).

An alternative proof is by induction on the number of elements in \( S \). \( \square \)

Homework 2

Problem 1.38

Here we had to show that

- all-NFAs recognize the class of regular languages
- NFAs recognize the class of languages defined by all-NFAs
Suppose $A$ is a regular language. This means that there is a DFA $D$ such that $L(D) = A$. But this means that every branch of the DFA accepts any string in $A$. Thus this DFA can be run as an all-NFA for $A$.

Suppose $A$ is a language such that there is an all-NFA $N$ such that $L(N) = A$. We need to show that we can construct an equivalent NFA (or DFA) that also recognizes this language. The issue here is that if we simply run the all-NFA as an NFA there will be strings not in $A$ that could be accepted as well. We will simulate this all-NFA by an equivalent DFA. The construction is very similar to the construction of an equivalent DFA from an NFA.

Given all-NFA $N = (Q, \Sigma, \delta, q_0, F)$ define a DFA $D = (Q', \Sigma, \delta', q'_0, F')$ where, as before, $Q' = \mathcal{P}(Q)$, $\delta'$ is the epsilon-closure of transitions from the elements of the state-set, $q'_0 = \{q_0\}$. However, the set of accept states $F' = \mathcal{P}(F)$. This is because the DFA should accept only when the simulated NFA ends all its branches in accept states. In other words, every branch ends in some $q_f \in F$.

**Problem 1.42**

This problem asks you to show that the class of regular languages is closed under shuffle. Suppose you are given regular languages $A$ and $B$. This means that DFAs $D_A = (Q_A, \Sigma, \delta_A, q_0^A, F_A)$ and $D_B = (Q_B, \Sigma, \delta_B, q_0^B, F_B)$ exist so that $L(D_A) = A$ and $L(D_B) = B$. Define a new NFA $N = (Q, \Sigma, \delta, q_0, F)$ that accepts strings that are in the shuffle of $A$ and $B$ by nondeterministically choosing to simulate one of the DFAs on each symbol in the input string.

That is, the set of states of $N$ is the cross product of the states of $D_A$ and $D_B$, $Q = Q_A \times Q_B$. The start state of $N$ is the pair of start states of $D_A$ and $D_B$, $q'_0 = (q_0^A, q_0^B)$. The accept states of $N$ is the pair of states where the first element is an accept state of $D_A$ and the second element is an accept state of $D_B$, $F = \{(q_{f_A}, q_{f_B}) \mid q_{f_A} \in F_A \text{ and } q_{f_B} \in F_B\}$. On input symbol $x \in \Sigma$, $\delta((q_A, q_B), x) = \{\delta_A(q_A, x), q_B), (q_A, \delta_B(q_B, x))\}$. That is the NFA $N$ nondeterministically simulates either the DFA $D_A$ or the DFA $D_B$.

**Homework 3**

**Problem 1.46(c)**

Let $A = \{w \mid w \in \{0, 1\}^* \text{ is not a palindrome}\}$.

Consider the complement of $A$, the language $\bar{A} = \{w \mid w \in \{0, 1\}^* \text{ is a palindrome}\}$. Since the class of regular languages is closed under complementation, it suffices to show that $\bar{A}$ is not regular. Suppose for a contradiction that $\bar{A}$ is regular. Consider the string $s = 0^p1^p \in \bar{A}$. Clearly, $|s| > p$. The pumping lemma for regular languages says that we can write $s = xyz$ so that $xy^iz \in \bar{A}$ for all $i \geq 0$ with $|y| > 0$ and $|xy| \leq p$. Since $|xy| \leq p$ and $|y| > 0$, it must be the case that $y = 0^k$ for some $k > 0$. But then $xz \notin \bar{A}$ since it has the form $0^j1^p$ for some $j < p$. This is a contradiction. Hence, $\bar{A}$ is not regular and neither is $A$. 
Problem 1.53
Consider the string \( 1^p = 0 + 1^p \in ADD \) and use an argument similar to the previous problem.

Homework 4: Practice Problems

Problem 2.6(b)
To define a CFG for the complement of the language \( A = \{a^n b^n | n \geq 0\} \) we first split the language \( \bar{A} \) into 3 components \( A_1 = \{a^i b^j | i > j\} \), \( A_2 = \{a^i b^j | i < j\} \) and \( A_3 = \Sigma^* b \Sigma^* a \Sigma^* \) so that \( \bar{A} = A_1 \cup A_2 \cup A_3 \).

The grammar for \( A_1 \) can be written as
\[
S_1 \rightarrow aS_1 b \mid aS_1 \mid a
\]
The grammar for \( A_2 \) can be written as
\[
S_2 \rightarrow aS_2 b \mid S_2 b \mid b
\]
The grammar for \( A_3 \) can be written as
\[
S_3 \rightarrow XbXaX \\
X \rightarrow aX \mid bX \mid \epsilon
\]
The grammar for \( \bar{A} \) can be written as
\[
S \rightarrow S_1 \mid S_2 \mid S_3
\]

Note on Practice Problem 1.54(c)
This problem shows that there are languages that are not regular that still satisfy the conditions of the pumping lemma. This is not a contradiction because the statement is that the pumping lemma necessarily holds for all regular languages.

It is important to be clear that the following two statements are not equivalent:

- If a language is regular then the pumping lemma holds
- If the pumping lemma holds for a language then it is regular