Homework Problems

1. [15 POINTS] Problem 22.14(a)-(c):
   Answer TRUE or FALSE.
   
   (a) A bijection must be an injection.
   **Solution:** True. Something is a bijection if and only if it is both an injection and a surjection.
   
   (b) There is a bijection from \(\mathbb{Q}\) to \(\mathbb{R}\).
   **Solution:** False. If there is a bijection between two sets, they have the same cardinality. \(\mathbb{Q}\) is countable, but \(\mathbb{R}\) is not.
   
   (c) There is a bijection from \(\mathbb{Q}\) to \(\mathbb{Z}\).
   **Solution:** True. Since both \(\mathbb{Q}\) and \(\mathbb{Z}\) are countable, there is a bijection between each of them and \(\mathbb{N}\). Then, it is possible to create a bijection from \(\mathbb{Q}\) to \(\mathbb{Z}\) by composing these two bijections: the bijection from \(\mathbb{Q}\) to \(\mathbb{Z}\) maps some \(q\) to some \(z\) iff the bijection from \(\mathbb{Q}\) to \(\mathbb{N}\) maps \(q\) to some \(n\) and the bijection from \(\mathbb{Z}\) to \(\mathbb{N}\) maps \(z\) to that same \(n\).

2. [15 POINTS] Problem 22.18(a)-(c):
   Prove or disprove.
   
   (a) \(\mathbb{Z}^2\) is the set of pairs \(\{(z_1, z_2) | z_1, z_2 \in \mathbb{Z}\}\). \(\mathbb{Z}^2\) is countable.
   **Solution:** True. Consider a Cartesian coordinate system. Enumerate pairs of integers by tracing a spiral on the Cartesian plane, starting at the origin, leading to the enumeration \((0, 0), (1, 0), (1, 1), (0, 1), (−1, 1), (−1, 0), (−1, −1), \ldots\) This listing will eventually contain every ordered pair of integers, thus \(\mathbb{Z}^2\) is countable.
   
   (b) \(\mathbb{Q}\) is the set of rational numbers, \(\mathbb{Q} = \{r | r = \frac{a}{b}, a \in \mathbb{Z}, \text{ and } b \in \mathbb{N}\}\). \(\mathbb{Q}\) is countable.
   **Solution:** True. Establish a mapping from \(\mathbb{Z}^2\) to \(\mathbb{Q}\) as follows: for every \((a, b) \in \mathbb{Z}^2\), map it to 0 if \(b = 0\) and \(\frac{a}{b}\) otherwise. Since every rational number is the quotient of two integers, every rational number has some pair of integers mapped to it, and thus this is a surjection. Since there exists a surjection from \(\mathbb{Z}^2\) to \(\mathbb{Q}\), \(\mathbb{Q}\) must have lesser or equal cardinality to \(\mathbb{Z}^2\). Since \(\mathbb{Z}^2\) is countable, as proven in (a), anything with equal or lesser cardinality is also countable, and thus \(\mathbb{Q}\) is countable.
   
   (c) \(F\) is the set of all functions from \(\mathbb{N}\) to \(\mathbb{N}\), \(F = \{f | f : \mathbb{N} \rightarrow \mathbb{N}\}\). \(F\) is countable.
   **Solution:** False, which can be shown with a variation of the diagonalization argument. Suppose that there is some enumeration \(f_1, f_2, \ldots\) of \(F\). Now, consider the function \(f(x) = f_x(x) + 1\). Since the enumeration contains all functions from integers to integers, and \(f\) is a function from integers to integers, there must exist some \(i\) such that \(f = f_i\). However, \(f(i) = f_i(i) + 1\), meaning that \(f(i) = f_i(i) = f_i(i) + 1\), which is a contradiction. Thus there cannot exist an enumeration of all functions from integers to integers, so \(F\) is uncountable.
3. [15 POINTS] Problem 24.5(d)-(f):
Give DFAs for the following computing problems.

(d) The strings which begin with 10 and end with 01.
Solution:
The above expression constitutes a language \( L \) in which all strings begin with 10 and end with 01. Note that the shortest accepted string in this scenario is 101.

(e) \( L = \{1^{2n}01^{2k+1}|n, k \geq 0\} \).
Solution:
Since both \( n, k \) can be zero, the shortest string here is 01.

(f) The language with all strings whose length is divisible by 3.
Solution:
A language \( L \) of all strings divisible by 3 requires that the length of all strings in \( L \) be \( 3k \), with \( k \geq 0 \). Therefore, the DFA always needs to consume sets of three characters to accept the string. Note that the empty string is also accepted in \( L \) (therefore, state \( q_0 \) is a final state).
4. [15 POINTS] Problem 25.4(a)-(c):
Give a DFA and a CFG for each computation problem below.

(a) \( L = \{01^n | n \geq 0 \} \)

Solution:
Language \( L \) defines a language of strings that begin with 0 and then have zero or more occurrences of 1 thereafter. The CFG and DFA are shown below.

\[
S \rightarrow 0A \\
A \rightarrow 1A | \varepsilon
\]
(b) \( \mathcal{L} = \{0^n1^n \mid 0 \leq n \leq 5 \} \)

**Solution:**

At first glance, one might think that \( \mathcal{L} \) is not solvable by a DFA, since in the absence of a constraint on \( n \), language \( \mathcal{L} \) requires memory; however, given the constraint on \( n \), we can enumerate all possible strings accepted by this language and design a CFG and DFA accordingly.

For the CFG, we might start with \( S \to 0S1 | \varepsilon \). From this pattern, constraining \( n \) to be no more than 5, we have:

\[
\begin{align*}
S & \to 0A1 | \varepsilon \\
A & \to 0B1 | \varepsilon \\
B & \to 0C1 | \varepsilon \\
C & \to 0D1 | \varepsilon \\
D & \to 01
\end{align*}
\]

For the DFA, we have the following (note that since each state must have a transition for 0 and for 1, the state transitions not shown all go to state \( E \)):
(c) $\mathcal{L} = \{\text{strings which end in a } 1\}$

**Solution:**
Language $\mathcal{L}$ has a very relaxed constraint in which we only care about whether the string ends in a 1; the rest of the sequence can be of any arbitrary binary pattern. The CFG and DFA are shown below.

$$S \to A1$$
$$A \to 0A \mid 1A \mid \varepsilon$$

![DFA Diagram]
5. **[15 POINTS] Problem 25.15(a)-(c):**

Consider the language \( \mathcal{L} = \{ \varepsilon, 1, 11, 111, \ldots \} = \{1\}^* \).

(a) Show that the CFG \( S \to \varepsilon | 1 | 1S \) generates \( \mathcal{L} \). Give a derivation of 111.

**Solution:**
Here we need to show that every string in \( \mathcal{L} \) is generated by the given CFG. We can use structural induction and assume the rules of \( S \) to prove our claim that \( S \to 1^k, k \geq 0 \).

**Base case:** For \( k = 0, w_0 = \varepsilon \). We need to show that \( S \to \varepsilon | 1 | 1S \) derives this. Quite simply, \( \varepsilon \) can be derived via \( S \to \varepsilon \). The base case holds.

**Inductive step:** We assume that for every string of length \( j \leq n \), the rules of the given CFG derive \( 1^j \). We then need to prove that every string of length \( j + 1 \) is derivable by the CFG. From the CFG rule \( S \to 1S \), we can obtain the string of length \( j + 1 \) from this rule. Here, we are only concatenating a 1, which forms a string also in \( \mathcal{L} \), thus we have shown that every string of length \( j + 1 \) is derivable by the given CFG.

Finally, the derivation of 111 is \( S \to 1S \to 11S \to 111 \). There are other possible derivations here.

(b) Show that the CFG \( S \to \varepsilon | 1 | SS \) generates \( \mathcal{L} \). Give two different derivations of 111.

**Solution:** Similar to the solution for (a) above, note that the CFG rule \( S \to SS \) (with the \( S \to 1 \) and \( S \to \varepsilon \) rules) will yield strings that then have zero, one, or two 1’s at each step of the derivation. You can then use induction to prove that the CFG generates \( \mathcal{L} \).

Two different derivations of 111 are:

1. \( S \to SS \to S1 \to SS1 \to S11 \to 111 \)
2. \( S \to SS \to 1S \to 1SS \to 11S \to 111 \)

Many other derivations exist for the string 111.

(c) A **leftmost** (rightmost) derivation replaces the leftmost (rightmost) variable at every step. For the grammar in (b), give leftmost and rightmost derivations of 111.

**Solution:** The leftmost and rightmost derivations of 111 are:

1. \( S \to SS \to 1S \to 1SS \to 11S \to 111 \)
2. \( S \to SS \to S1 \to SS1 \to S11 \to 111 \)
6. [15 POINTS] Problem 25.16(a)-(c):
Given the following CFG: \( S \rightarrow A1B; A \rightarrow \varepsilon|0A; B \rightarrow \varepsilon|0B|1B \). Give leftmost and rightmost derivations of the following. Also, give parse trees for each of your derivations.

(a) 00101

**Solution:** Leftmost derivation is:
\[
S \rightarrow A1B \rightarrow 0A1B \rightarrow 00A1B \rightarrow 001B \rightarrow 0010B \rightarrow 00101B \rightarrow 00101.
\]
The leftmost parse tree is (the red numbers are optional, but help to show each step):

Rightmost derivation is:
\[
S \rightarrow A1B \rightarrow A10B \rightarrow A101B \rightarrow A101 \rightarrow 0A101 \rightarrow 00A101 \rightarrow 00101.
\]
The rightmost parse tree is (the red numbers are optional, but help to show each step):
(b) 1001

**Solution:** Leftmost derivation is:

\[ S \rightarrow A1B \rightarrow 1B \rightarrow 10B \rightarrow 100B \rightarrow 1001B \rightarrow 1001. \]

The leftmost parse tree is (the red numbers are optional, but help to show each step):

```
          S
          /\  \
         /   \  
        A     B
        /\  /\  \
       /  \\ /  \
      /    0  \
     /      /  \
    /       4  \
   /         /  \
  /          0  \
 /           /  \\  
/            5\
/              /  \
/                6\
/                  /  \\  
/                   6\
/                     /  \
/                      /  \\  
/                        6\
/                          /  \\  
/                             6\
/                               /  \
/                                 /  \\  
/                                   6\
/                                     /  \
/                                       /  \\  
/                                         6\
/                                           /  \
/                                             /  \\  
/                                               6\
/                                                 /  \
/                                                   /  \\  
/                                                     6\
/                                                       /  \
/                                                         /  \\  
/                                                           6\
/                                                               /  \
/                                                                 /  \\  
/                                                                     6\
/                                                                               /  \
/                                                                                       /  \\  
/                                                                                                 6\
```

Rightmost derivation is:

\[ S \rightarrow A1B \rightarrow A10B \rightarrow A100B \rightarrow A1001B \rightarrow A1001 \rightarrow 1001. \]

The rightmost parse tree is (the red numbers are optional, but help to show each step):

```
          S
          /\  \
         /   \  
        A     B
        /\  /\  \
       /  \\ /  \
      /    0  \
     /      /  \
    /       4  \
   /        /  \\  
  /        0     \
 /         /  \\  
/          5\
/           /  \\  
/             5\
/               /  \\  
/                 5\
/                   /  \\  
/                     5\
/                       /  \\  
/                           5\
```

8
(c) 00011

**Solution:** Leftmost derivation is:

\[ S \rightarrow A1B \rightarrow 0A1B \rightarrow 00A1B \rightarrow 000A1B \rightarrow 0001B \rightarrow 00011B \rightarrow 00011. \]

The leftmost parse tree is (the red numbers are optional, but help to show each step):

![Leftmost Parse Tree](image)

Rightmost derivation is:

\[ S \rightarrow A1B \rightarrow A11B \rightarrow A11 \rightarrow 0A11 \rightarrow 00A11 \rightarrow 000A11 \rightarrow 00011. \]

The rightmost parse tree is (the red numbers are optional, but help to show each step):

![Rightmost Parse Tree](image)
7. [10 POINTS] Problem 26.4(a)-(b):
Give high-level pseudocode for Turing Machines that solve these problems. In some cases you
are asked for a decider. In other cases, you are asked for a transducer.

(a) Regular language: \( L_1 = \{01^*\} \) and \( L_2 = \{01\} \).

Solution: These are two decider problems. For \( L_1 \), the Turing Machine would move
into an accept state once a consecutive 01 is detected on the input. For \( L_2 \), the Turing
Machine would be similar to that of \( L_1 \), except the accept state must be 01_ (i.e., no
other input may follow).

For the Turing Machine that accepts \( L_1 \) (i.e., \( TM(L_1) \)), see the solution to Exer-
cise 26.6(a). Next, for the Turing Machine that accepts \( L_2 \) (i.e., \( TM(L_2) \)), modify this
solution by changing accept state \( A \) to be state \( q_2 \); from state \( q_2 \), transition \( \{\} \{\} \{R\} \)
leads to new accept state \( A \). Also, transitions back to previous states and error state \( E \)
are required here.

(b) Not CFL: \( L = \{0^n\#1^n\#0^n\} \) (\( \# \) is a ‘punctuation’ symbol).

Solution: Since this is not a CFL, only a Turing Machine can solve this computing
problem. Note that like (a), this is also a decider.

At a high level, the pseudocode is as follows:

i. Check the format to ensure that there are two \( \# \) symbols. Accomplish this by
moving to the right until the first \( \# \) symbol is encountered, then moving right until
the second \( \# \) symbol is encountered. If at any point, the end of the input (i.e., \( \empt \)) is
encountered, transition to the error state. Note that this can be accomplished by a
simple DFA.

ii. Essentially, for each 0 encountered before the first \( \# \) symbol:

1. Mark the 0 (to remember where you left off)
2. Move right past the first \( \# \) symbol until the first unmarked 1 is encountered.
   Mark the 1, then move right past the second \( \# \) symbol until the first unmarked
   0 is encountered. Mark this unmarked 0. (If at any point, the desired symbol is
   not found, enter an error state.)
3. Next, move left past the two \( \# \) symbols, finding the rightmost marked 0, then
   move right. If a \( \# \) symbol is encountered, goto step 4. If a 0 is encountered, go
to step 1. Otherwise, reject.
4. Move right until the end of the input (i.e., \( \empt \)) is encountered. If no unmarked 0
   or 1 symbols were encountered, accept; otherwise reject.