1. Structural Induction I

Consider the following recursive definition of strings $\Sigma^*$ over the alphabet $\Sigma$.

**Basis Step:** $\varepsilon$ is a string

**Recursive Step:** If $w$ is a string and $a \in \Sigma$ is a character, then $wa$ is a string.

Recall the following recursive definition of the function $\text{len}$:

\[
\text{len}(\varepsilon) = 0 \\
\text{len}(wa) = 1 + \text{len}(w)
\]

Now, consider the following recursive definition:

\[
\text{double}(\varepsilon) = \varepsilon \\
\text{double}(wa) = \text{double}(w)aa.
\]

Prove that, for any string $x$, we have $\text{len} \left( \text{double}(x) \right) = 2 \cdot \text{len}(x)$.

**Solution:**

Let $P(x)$ be \("\text{len} \left( \text{double}(x) \right) = 2 \cdot \text{len}(x)\"\). We prove $P(x)$ for all strings $x \in \Sigma^*$ by structural induction.

**Base Case.** By definition, $\text{len} \left( \text{double}(\varepsilon) \right) = \text{len} \left( \varepsilon \right) = 0 = 2 \cdot 0 = 2 \cdot \text{len}(\varepsilon)$, so $P(\varepsilon)$ holds.

**Induction Hypothesis.** Suppose $P(w)$ holds for some arbitrary string $w$.

**Induction Step.** We show that $P(wa)$ holds, for any character $a \in \Sigma$, as follows:

\[
\text{len} \left( \text{double}(wa) \right) = 2 \cdot \text{len}(wa) \\
= 2 + 2 \cdot \text{len}(w) \\
= 2 \left( 1 + \text{len}(w) \right) \\
= 2 \text{len}(wa)
\]

Thus, $P(x)$ holds for all strings $x \in \Sigma^*$ by structural induction.

2. Structural Induction II

Consider the following definition of a (binary) Tree:

**Basis Step:** $\bullet$ is a Tree.

**Recursive Step:** If $L$ is a Tree and $R$ is a Tree then Tree($\bullet, L, R$) is a Tree.

The function leaves returns the number of leaves of a Tree. It is defined as follows:

\[
\text{leaves}(\bullet) = 1 \\
\text{leaves}(\text{Tree}(\bullet, L, R)) = \text{leaves}(L) + \text{leaves}(R)
\]

Also, recall the definition of size on trees:

\[
\text{size}(\bullet) = 1 \\
\text{size}(\text{Tree}(\bullet, L, R)) = 1 + \text{size}(L) + \text{size}(R)
\]

Prove that $\text{leaves}(T) \geq \frac{\text{size}(T)}{2}$ for all $T \in \text{Trees}$. 

Solution:
In this problem, we define a strengthened predicate. For a tree $T$, let $P$ be $\text{leaves}(T) \geq \text{size}(T)/2 + 1/2$. We prove $P$ for all trees $T$ by structural induction.

**Base Case.** We show that $P$ holds. By definition of $\text{leaves}(\cdot)$, $\text{leaves}(\bullet) = 1$ and $\text{size}(\bullet) = 1$. So, $\text{leaves}(\bullet) = 1 \geq 1/2 + 1/2 = \text{size}(\bullet)/2 + 1/2$.

**Induction Hypothesis:** Suppose $P(L)$ and $P(R)$ hold for some arbitrary trees $L$ and $R$.

**Induction Step:** We prove that $P(\text{Tree}(\bullet, L, R))$ holds as follows:

\[
\text{leaves(}\text{Tree}(\bullet, L, R)) = \text{leaves}(L) + \text{leaves}(R) \geq (\text{size}(L)/2 + 1/2) + (\text{size}(R)/2 + 1/2) = (\text{size}(L) + \text{size}(R) + 1)/2 + 1/2 = \text{size(}\text{Tree}(\bullet, L, R))/2 + 1/2.
\]

Thus, the $P(T)$ holds for all trees $T$.

3. **Regular Expressions**
   (a) Write a regular expression that matches base 10 non-negative numbers.
      (Note that there should be no leading zeroes.)

   **Solution:**
   \[
   0 \cup ((1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)^*)
   \]
   
   (b) Write a regular expression that matches all non-negative base-3 numbers that are divisible by 3.

   **Solution:**
   \[
   0 \cup ((1 \cup 2)(0 \cup 1 \cup 2)^*0)
   \]
   
   (c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

   **Solution:**
   \[
   (01 \cup 001 \cup 1^*)^*(0 \cup 00 \cup \varepsilon)111(01 \cup 001 \cup 1^*)^*(0 \cup 00 \cup \varepsilon)
   \]
   (If you don’t want the substring 000, the only way you can produce 0s is if there are only one or two 0s in a row, and they are immediately followed by a 1 or the end of the string.)
4. CFGs
Construct CFGs for the following languages:

(a) All binary strings that end in 00.

Solution:

\[ S \rightarrow 0S \mid 1S \mid 00 \]

(b) All binary strings that contain at least three 1’s.

Solution:

\[
\begin{align*}
S & \rightarrow TTT \\
T & \rightarrow 0T \mid T0 \mid 1T \mid 1
\end{align*}
\]

(c) Propositional logic statements using only variables from a fixed alphabet \( \mathcal{A} = \{ \ldots, p, q, r, \ldots \} \) and only the operators \( \neg, \land, \lor \) as well as parentheses "(\()". (Assume no space characters.)

Solution:

\[
\begin{align*}
S & \rightarrow F \mid S \lor F \\
F & \rightarrow P \mid F \land F \\
P & \rightarrow V \mid (S) \mid \neg P \\
V & \rightarrow \cdots \mid p \mid q \mid r \mid \ldots
\end{align*}
\]

Note that this gives \( \land \) higher precedence than \( \lor \), as would be expected.

5. Structural Induction III
In this problem, we will prove De Morgan’s Law for arbitrary propositions. For example, we will show that

\[-(p_1 \land p_2 \land \cdots \land p_n) \equiv \neg p_1 \lor \neg p_2 \lor \cdots \lor \neg p_n.\]

is true for any \( n \geq 1 \).

Let \( \mathcal{A} = \{ \ldots, p, q, r, \ldots \} \) be a fixed set of atomic propositions. We then define the set \( \text{Prop} \) as follows:

**Basis Elements** For any \( p \in \mathcal{A} \), \( \text{Atomic}(p) \in \text{Prop} \).

**Recursive Step** If \( A, B \in \text{Prop} \), then \( \neg(A), \text{Wedge}(A, B), \text{Vee}(A, B) \in \text{Prop} \).

The set \( \text{Prop} \) represents parse trees of propositions. We allow the propositions to be combined using the operators, Wedge and Vee (the names of \( \land \) and \( \lor \) in \( \LaTeX \)). We also allow negation of propositions with Neg.

Next, we define a function \( T \) that takes a parse tree (an element of \( \text{Prop} \)) as input and returns the proposition that it represents. Formally we define,

\[
\begin{align*}
T(\text{Atomic}(p)) &= p & \text{for any } p \in \mathcal{A} \\
T(\text{Wedge}(A, B)) &= (T(A)) \land (T(B)) & \text{for any } A, B \in \text{Prop} \\
T(\text{Vee}(A, B)) &= (T(A)) \lor (T(B)) & \text{for any } A, B \in \text{Prop} \\
T(\neg(A)) &= \neg T(A) & \text{for any } A \in \text{Prop}
\end{align*}
\]
The function flip takes a parse tree as input and returns another parse tree as follows:

\[
\begin{align*}
\text{flip}(\text{Atomic}(p)) &= \text{Neg}(\text{Atomic}(p)) & \text{for any } p \in A \\
\text{flip}(\text{Wedge}(A, B)) &= \text{Vee}(\text{flip}(A), \text{flip}(B)) & \text{for any } A, B \in \text{Prop} \\
\text{flip}(\text{Vee}(A, B)) &= \text{Wedge}(\text{flip}(A), \text{flip}(B)) & \text{for any } A, B \in \text{Prop} \\
\text{flip}(\text{Neg}(A)) &= A & \text{for any } A \in \text{Prop}
\end{align*}
\]

The function flip negates each atomic proposition and swaps \( \lor \) with \( \land \) (and vice versa) throughout the tree.

With those definitions in hand, use structural induction show that, for any \( A \in \text{Prop} \),

\[
\mathcal{T}(\text{Neg}(A)) \equiv \mathcal{T}(\text{flip}(A)).
\]

This proves that we can produce a proposition that is equivalent to negating the expression by, instead, flipping all \( \land \)s to \( \lor \)s (and vice versa) and negating atomic propositions recursively until we hit \( \neg \)s.

**Solution:**

Let \( P(A) \) be “\( \mathcal{T}(\text{Neg}(A)) \equiv \mathcal{T}(\text{flip}(A)) \)”. We prove \( P(A) \) for all \( A \in \text{Prop} \) by structural induction.

**Base Case** Let \( p \) be an arbitrary member of \( A \). In this case, \( P(\text{Atomic}(p)) \) says

\[
\mathcal{T}(\text{Neg}(\text{Atomic}(p))) = \mathcal{T}(\text{flip}(\text{Atomic}(p))),
\]

which is immediate from the definition of flip (read right-to-left).

**Induction Hypothesis** Suppose \( P(A) \) and \( P(B) \) hold for some arbitrary \( A \) and \( B \) in \( \text{Prop} \).

**Induction Step** We show \( P(\text{Wedge}(A, B)) \) as follows (\( P(\text{Vee}(A, B)) \) is similar and left as an exercise):

\[
\begin{align*}
\mathcal{T}(\text{Neg}(\text{Wedge}(A, B))) &= \neg \mathcal{T}(\text{Wedge}(A, B)) & \text{Def of } \mathcal{T} \\
&= \neg (\mathcal{T}(A) \land \mathcal{T}(B)) & \text{Def of } \mathcal{T} \\
&\equiv \neg \mathcal{T}(A) \lor \neg \mathcal{T}(B) & \text{De Morgan's Law} \\
&= \mathcal{T}(\text{Neg}(A)) \lor \mathcal{T}(\text{Neg}(B)) & \text{Def of } \mathcal{T} \\
&\equiv \mathcal{T}(\text{flip}(A)) \lor \mathcal{T}(\text{flip}(B)) & \text{Induction Hypothesis} \\
&= \mathcal{T}(\text{Vee}(\text{flip}(A), \text{flip}(B))) & \text{Def of } \mathcal{T} \\
&= \mathcal{T}(\text{flip}(\text{Wedge}(A, B))) & \text{Def of flip}
\end{align*}
\]

We can show \( P(\text{Neg}(A)) \) as follows:

\[
\begin{align*}
\mathcal{T}(\text{Neg}(\text{Neg}(A))) &= \neg \mathcal{T}(\text{Neg}(A)) & \text{Def of } \mathcal{T} \\
&= \neg \neg \mathcal{T}(A) & \text{Def of } \mathcal{T} \\
&\equiv \mathcal{T}(A) & \text{Double Negation} \\
&= \mathcal{T}(\text{flip}(\text{Neg}(A))) & \text{Def of flip}
\end{align*}
\]

Thus, \( P(A) \) holds for all parse trees \( A \in \text{Prop} \), by structural induction.