Section 07: Structural Induction, REs, and CFGs

1. Structural Induction

(a) Consider the following recursive definition of strings.

Basis Step: `""` is a string

Recursive Step: If X is a string and c is a character then $\mathsf{append}(c, X)$ is a string.

Recall the following recursive definition of the function len:

$$\begin{aligned} \mathsf{len}(\texttt{""}) &= 0 \\ \mathsf{len}(\mathsf{append}(c, X)) &= 1 + \mathsf{len}(X) \end{aligned}$$

Now, consider the following recursive definition:

$$\begin{aligned} \mathsf{double}(\texttt{""}) &= \texttt{""} \\ \mathsf{double}(\mathsf{append}(c, X)) &= \mathsf{append}(c, \mathsf{append}(c, \mathsf{double}(X))). \end{aligned}$$

Prove that for any string X, $\mathsf{len}(\mathsf{double}(X)) = 2\mathsf{len}(X)$.

(b) 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 $\texttt{Tree}(\bullet, L, R)$ is a **Tree**.

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

$$\begin{aligned} \mathsf{leaves}(\bullet) &= 1 \\ \mathsf{leaves}(\texttt{Tree}(\bullet, L, R)) &= \mathsf{leaves}(L) + \mathsf{leaves}(R) \end{aligned}$$

Also, recall the definition of size on trees:

$$\begin{aligned} \mathsf{size}(\bullet) &= 1 \\ \mathsf{size}(\texttt{Tree}(\bullet, L, R)) &= 1 + \mathsf{size}(L) + \mathsf{size}(R) \end{aligned}$$

Prove that $\mathsf{leaves}(T) \geq \mathsf{size}(T)/2 + 1/2$ for all Trees T.

(c) Prove the previous claim using strong induction. Define $P(n)$ as "all trees T of size n satisfy $\mathsf{leaves}(T) \geq \mathsf{size}(T)/2 + 1/2$". You may use the following facts:

- For any tree T we have $\mathsf{size}(T) \geq 1$.
- For any tree T, $\mathsf{size}(T) = 1$ if and only if $T = \bullet$.

If we wanted to prove these claims, we could do so by structural induction.

Note, in the inductive step you should start by letting T be an arbitrary tree of size $k + 1$.

2. Regular Expressions

(a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

(b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

(c) Write a regular expression that matches all binary strings that contain the substring "111", but not the substring "000".

3. CFGs

(a) All binary strings that end in 00.

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

(c) All strings over $\{0,1,2\}$ with the same number of 1s and 0s and exactly one 2.
Hint: Try modifying the grammar from lecture for binary strings with the same number of 1s and 0s. (You may need to introduce new variables in the process.)

4. Walk the Dawgs

Suppose a dog walker takes care of $n \geq 12$ dogs. The dog walker is not a strong person, and will walk dogs in groups of 3 or 7 at a time (every dog gets walked exactly once). Prove the dog walker can always split the n dogs into groups of 3 or 7.

5. Reversing a Binary Tree

Consider the following definition of a (binary) **Tree**.

Basis Step `Nil` is a **Tree**.

Recursive Step If L is a **Tree**, R is a **Tree**, and x is an integer, then $\texttt{Tree}(x, L, R)$ is a **Tree**.

The sum function returns the sum of all elements in a **Tree**.

$$\begin{aligned} \mathsf{sum}(\texttt{Nil}) &= 0 \\ \mathsf{sum}(\texttt{Tree}(x, L, R)) &= x + \mathsf{sum}(L) + \mathsf{sum}(R) \end{aligned}$$

The following recursively defined function produces the mirror image of a **Tree**.

$$\begin{aligned} \mathsf{reverse}(\texttt{Nil}) &= \texttt{Nil} \\ \mathsf{reverse}(\texttt{Tree}(x, L, R)) &= \texttt{Tree}(x, \mathsf{reverse}(R), \mathsf{reverse}(L)) \end{aligned}$$

Show that, for all **Tree**s T that

$$\mathsf{sum}(T) = \mathsf{sum}(\mathsf{reverse}(T))$$