Section 07: Structural Induction, Functions, Regular Expressions, CFGs

1. Recursively Defined Sets of Strings

For each of the following, write a recursive definition of the sets satisfying the following properties. You do not need to mention the exclusion rule. Briefly justify that your solution is correct.

(a) Binary strings of even length.

(b) Binary strings not containing 10.

(c) Binary strings not containing 10 as a substring and having at least as many 1s as 0s.

2. 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.

3. 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

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

4. One-to-One and Onto

Determine if the following functions are (1) one-to-one and (2) onto. For each claim: provide a proof if true, otherwise give a counterexample (you must also explain why the counterexample works).

(a) $f : \mathbb{N} \to \mathbb{N}$, $f(x) = x^2$

(b) $f : \mathbb{R} \to \mathbb{R}$, $f(x) = x^2$

(c) $f : \mathbb{R}^+ \to \mathbb{R}^+$, $f(x) = x^2$, where $\mathbb{R}^+ = \{x : x \in \mathbb{R} \wedge x \geq 0\}$, i.e., the set of non-negative real numbers.

Notice that the domain and co-domain matter! You have to know both to tell whether the function is one-to-one or onto.

5. A Bijection Proof

Let A be the set of negative integers, i.e., $A = \{-1, -2, -3, ...\}$; let B be the set of integers at least 10, i.e., $B = \{10, 11, 12, 13, ...\}$ Show that $f : A \to B$ defined by $f(x) = |x| + 9$ is a bijection.

You may use these facts:

- for negative numbers x, y: $|x| = |y| \to x = y$
- for negative numbers $|x| = -x$

that

6. 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 (again, there should be no leading zeroes.).

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

(d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

(e) Write a regular expression that matches all binary strings of the form $1^k y$, where $k \geq 1$ and $y \in \{0,1\}^*$ has at least k 1's.

7. CFGs

Write a context-free grammar to match each of these languages.

(a) All binary strings that start with 11.

(b) All binary strings that contain at most one 1.

(c) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.
Hint: First try writing a grammar that just produces binary strings with the same number of 1s and 0s (i.e., doesn't have the "exactly one 2 requirement".