

Section 07: Structural Induction, Regular Expressions, Context-Free Grammars

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 $\text{append}(c, X)$ is a string.

Recall the following recursive definition of the function len :

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

Now, consider the following recursive definition:

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

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

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

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

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

Also, recall the definition of size on trees:

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

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

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

- For any tree T we have $\text{size}(T) \geq 1$.
- For any tree T , $\text{size}(T) = 1$ 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 binary strings with an equal number of 1's and 0's.

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 $\text{Tree}(x, L, R)$ is a **Tree**.

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

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

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

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

Show that, for all **Trees** T that

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

6. Recursively Defined Sets of Strings

For each of the following, write a recursive definition of the sets satisfying the following properties. 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.
- (d) Binary strings containing at most two 0s and at most two 1s.