# CSE 311: Foundations of Computing I

## Section 8: Structural Induction and Regular Expressions Solutions

## 1. Structural Induction

(a) 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 len:

$$\begin{aligned} \mathsf{len}(\varepsilon) &= 0 \\ \mathsf{len}(wa) &= 1 + \mathsf{len}(w) \end{aligned}$$

Now, consider the following recursive definition:

$$\begin{aligned} \mathsf{double}(\varepsilon) &= \varepsilon \\ \mathsf{double}(wa) &= \mathsf{double}(w)aa. \end{aligned}$$

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

### Solution:

For a string $x$, let $\mathsf{P}(x)$ be "$\mathsf{len}(\mathsf{double}(x)) = 2\mathsf{len}(x)$. We prove $\mathsf{P}(x)$ for all strings $x \in \Sigma^*$ by structural induction.

**Base Case.** We show $\mathsf{P}(\varepsilon)$ holds. By definition $\mathsf{len}(\mathsf{double}(\varepsilon)) = \mathsf{len}(\varepsilon) = 0 = 2\mathsf{len}(\varepsilon)$, as desired.

**Induction Hypothesis.** Suppose $\mathsf{P}(w)$ holds for some arbitrary string $w$.

**Induction Step.** We show that $\mathsf{P}(wa)$ holds for any character $a \in \Sigma$.

$$\begin{aligned} \mathsf{len}(\mathsf{double}(wa)) &= \mathsf{len}(\mathsf{double}(w)aa) && \text{[By Definition of double]} \\ &= 1 + \mathsf{len}(\mathsf{double}(w)a) && \text{[By Definition of len]} \\ &= 1 + 1 + \mathsf{len}(\mathsf{double}(w)) && \text{[By Definition of len]} \\ &= 2 + 2\mathsf{len}(w) && \text{[By IH]} \\ &= 2(1 + \mathsf{len}(w)) && \text{[Algebra]} \\ &= 2(\mathsf{len}(wa)) && \text{[By Definition of len]} \end{aligned}$$

This proves $\mathsf{P}(wa)$.

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

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

**Solution:**

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

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

$$\begin{aligned}
\text{leaves}(\texttt{Tree}(\bullet, L, R)) &= \text{leaves}(L) + \text{leaves}(R) && \text{[By Definition of leaves]} \\
&\geq (\text{size}(L)/2 + 1/2) + (\text{size}(R)/2 + 1/2) && \text{[By IH]} \\
&= (\text{size}(L) + \text{size}(R) + 1)/2 + 1/2 \\
&= \text{size}(\texttt{Tree}(\bullet, L, R))/2 + 1/2 && \text{[By Definition of size]}
\end{aligned}$$

This proves P$(\texttt{Tree}(\bullet, L, R))$.

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

## 2. Regular Expressions

(a) Write a regular expression that matches base 10 non-negative numbers (e.g., 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.)