Section 08: Solutions

1. Regular Expressions

(a) Write a regular expression that matches base 10 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 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:

$$(01 \cup 001 \cup 1^*)^*(0 \cup 00 \cup \varepsilon)111(01 \cup 001 \cup 1^*)^*(0 \cup 00 \cup \varepsilon)$$

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

Solution:

$$((01)^*(0 \cup \varepsilon)) \cup ((10)^*(1 \cup \varepsilon))$$

(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.

Solution:

$$1(0 \cup 1)^* 1(0 \cup 1)^*$$

Explanation: While it may seem like we need to keep track of how many 1's there are, it turns out that we don't. Convince yourself that strings in the language are exactly those of the form $1x$, where x is any binary string with at least one 1. Hence, x is matched by the regular expression $(0 \cup 1)^* 1(0 \cup 1)^*$.

2. CFGs

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

(a) All binary strings that end in 00.
Solution:

$$\mathbf{S} \to 0\mathbf{S} \mid 1\mathbf{S} \mid 00$$

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

$$\mathbf{S} \to \mathbf{TTT}$$
$$\mathbf{T} \to 0\mathbf{T} \mid \mathbf{T}0 \mid 1\mathbf{T} \mid 1$$

(c) All binary strings with an equal number of 1's and 0's.
Solution:

$$\mathbf{S} \to 0\mathbf{S}1\mathbf{S} \mid 1\mathbf{S}0\mathbf{S} \mid \varepsilon$$

and

$$\mathbf{S} \to \mathbf{SS} \mid 0\mathbf{S}1 \mid 1\mathbf{S}0 \mid \varepsilon$$

both work. Note: The fact that all the strings generated have the property is easy to show (by induction) but the fact that one can generate all strings with the property is trickier. To argue this that each of these is grammars is enough one would need to consider how the difference between the # of 0's seen and the # of 1's seen occurs in prefixes of any string with the property.

(d) All binary strings of the form xy, where $|x| = |y|$, but $x \neq y$.

Solution:

$$\mathbf{S} \to \mathbf{AB} \mid \mathbf{BA}$$
$$\mathbf{A} \to 0 \mid 0\mathbf{A}0 \mid 0\mathbf{A}1 \mid 1\mathbf{A}0 \mid 1\mathbf{A}1$$
$$\mathbf{B} \to 1 \mid 0\mathbf{B}0 \mid 0\mathbf{B}1 \mid 1\mathbf{B}0 \mid 1\mathbf{B}1$$

Explanation: We will explain the forward direction (i.e. this grammar generates strings of the desired form); in particular, we will examine strings generated by the rule **AB**, as the other rule follows similarly. An arbitrary string generated by **AB** will look like $a_1 0 a_2 b_1 1 b_2$, where $a_1, a_2, b_1, b_2 \in \{0,1\}^*$, $|a_1| = |a_2| = k_1$, and $|b_1| = |b_2| = k_2$ for some $k_1, k_2 \in \mathbb{N}$. In particular, we can "repartition" the substring $a_2 b_1$ into $a_2' b_1'$ s.t. $|a_2'| = k_2$ and $|b_1'| = k_1$. Letting $x = a_1 0 a_2'$ and $y = b_1' 1 b_2$, observe that $|x| = |y| = k_1 + k_2 + 1$ and x and y differ at the (k_1+1)-th character.

3. 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, $\mathsf{len}(\mathsf{double}(X)) = 2\mathsf{len}(X)$.

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 by structural induction on X.

Base Case (X = `""`): By definition, $\mathsf{len}(\mathsf{double}(\texttt{""})) = \mathsf{len}(\texttt{""}) = 0 = 2 \cdot 0 = 2\mathsf{len}(\texttt{""})$, so $\mathsf{P}(\texttt{""})$ holds

Inductive Hypothesis: Suppose $\mathsf{P}(X)$ holds for some arbitrary string X.

Inductive Step: $\boxed{\text{Goal: Show that } \mathsf{P}(\mathsf{append}(c, X)) \text{ holds for any character } c.}$

$$\begin{aligned} \mathsf{len}(\mathsf{double}(\mathsf{append}(c, X))) &= \mathsf{len}(\mathsf{append}(c, \mathsf{append}(c, \mathsf{double}(X)))) && \text{[By Definition of double]} \\ &= 1 + \mathsf{len}(\mathsf{append}(c, \mathsf{double}(X))) && \text{[By Definition of len]} \\ &= 1 + 1 + \mathsf{len}(\mathsf{double}(X)) && \text{[By Definition of len]} \\ &= 2 + 2\mathsf{len}(X) && \text{[By IH]} \\ &= 2(1 + \mathsf{len}(X)) && \text{[Algebra]} \\ &= 2(\mathsf{len}(\mathsf{append}(c, X))) && \text{[By Definition of len]} \end{aligned}$$

This proves $\mathsf{P}(\mathsf{append}(c, X))$.

Conclusion: $\mathsf{P}(X)$ holds for all strings X 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 $\mathsf{leaves}(T) \geq \mathsf{size}(T)/2 + 1/2$ for all Trees T.

Solution:

For a tree T, let P be $\mathsf{leaves}(T) \geq \mathsf{size}(T)/2 + 1/2$. We prove P for all trees T by structural induction on T.

Base Case (T = $\bullet$): By definition of $\mathsf{leaves}(\bullet)$, $\mathsf{leaves}(\bullet) = 1$ and $\mathsf{size}(\bullet) = 1$. So, $\mathsf{leaves}(\bullet) = 1 \geq 1/2 + 1/2 = \mathsf{size}(\bullet)/2 + 1/2$, so $\mathsf{P}(\bullet)$ holds.

Inductive Hypothesis: Suppose $\mathsf{P}(L)$ and $\mathsf{P}(R)$ hold for some arbitrary trees L, R.

Inductive Step: $\boxed{\text{Goal: Show that } \mathsf{P}(\texttt{Tree}(\bullet, L, R)) \text{ holds.}}$

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

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

Conclusion: Thus, $\mathsf{P}(T)$ holds for all trees T by structural induction.

(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 $k + 1$.

Solution:

Let $P(n)$ be "all trees T of size n satisfy $\mathsf{leaves}(T) \geq \mathsf{size}(T)/2 + 1/2$". We show $P(n)$ for all integers $n \geq 1$ by strong induction on n.

Base Case: Let T be an arbitrary tree of size 1. The only tree with size 1 is $\bullet$, so $T = \bullet$. By definition, $\mathsf{leaves}(T) = \mathsf{leaves}(\bullet) = 1$ and thus $\mathsf{size}(T) = 1 = 1/2 + 1/2 = \mathsf{size}(T)/2 + 1/2$. This shows the base case holds.

Inductive Hypothesis: Suppose that $P(j)$ holds for all integers $j = 1, 2, \ldots, k$ for some arbitrary integer $k \geq 1$.

Inductive Step: Let T be an arbitrary tree of size $k + 1$. Since $k + 1 > 1$, we must have $T \neq \bullet$. It follows from the definition of a tree that $T = \texttt{Tree}(\bullet, L, R)$ for some trees L and R. By definition, we have $\mathsf{size}(T) = 1 + \mathsf{size}(L) + \mathsf{size}(R)$. Since sizes are non-negative, this equation shows $\mathsf{size}(T) > \mathsf{size}(L)$ and $\mathsf{size}(T) > \mathsf{size}(R)$ meaning we can apply the inductive hypothesis. This says that $\mathsf{leaves}(L) \geq \mathsf{size}(L)/2 + 1/2$ and $\mathsf{leaves}(R) \geq \mathsf{size}(R)/2 + 1/2$.

We have,

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

This shows $P(k + 1)$.

Conclusion: $P(n)$ holds for all integers $n \geq 1$ by the principle of strong induction.

Note, this proves the claim for all trees because every tree T has some size $s \geq 1$. Then $P(s)$ says that all trees of size s satisfy the claim, including T.

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.

Solution:

Let $P(n)$ be "a group with n dogs can be split into groups of 3 or 7 dogs." We will prove $P(n)$ for all natural numbers $n \geq 12$ by strong induction.

Base Cases $n = 12, 13, 14,$ **or** 15**:** $12 = 3 + 3 + 3 + 3$, $13 = 3 + 7 + 3$, $14 = 7 + 7$, So $P(12)$, $P(13)$, and $P(14)$ hold.

Inductive Hypothesis: Assume that $P(12), \ldots, P(k)$ hold for some arbitrary $k \geq 14$.

Inductive Step: $\boxed{\text{Goal: Show } k+1 \text{ dogs can be split into groups of size 3 or 7.}}$

We first form one group of 3 dogs. Then we can divide the remaining $k-2$ dogs into groups of 3 or 7 by the assumption $P(k-2)$. (Note that $k \geq 14$ and so $k-2 \geq 12$; thus, $P(k-2)$ is among our assumptions $P(12), \ldots, P(k)$.)

Conclusion: $P(n)$ holds for all integers $n \geq 12$ by by principle of strong induction.

5. For All

For this problem, we'll see an incorrect use of induction. For this problem, we'll think of all of the following as binary trees:

- A single node.
- A root node, with a left child that is the root of a binary tree (and no right child)
- A root node, with a right child that is the root of a binary tree (and no left child)
- A root node, with both left and right children that are roots of binary trees.

Let $P(n)$ be "for all trees of height n, the tree has an odd number of nodes"

Take a moment to realize this claim is false.

Now let's see an incorrect proof:

We'll prove $P(n)$ for all $n \in \mathbb{N}$ by induction on n.

Base Case ($n = 0$): There is only one tree of height 0, a single node. It has one node, and $1 = 2 \cdot 0 + 1$, which is an odd number of nodes.

Inductive Hypothesis: Suppose $P(i)$ holds for $i = 0, ..., k$, for some arbitrary $k \geq 0$.

Inductive Step: Let T be an arbitrary tree of height k. All trees with nodes (and since $k \geq 0$, T has at least one node) have a leaf node. Add a left child and right child to a leaf (pick arbitrarily if there's more than one), This tree now has height $k + 1$ (since T was height k and we added children below). By IH, T had an odd number of nodes, call it $2j + 1$ for some integer j. Now we have added two more, so our new tree has $2j + 1 + 2 = 2(j + 1) + 1$ nodes. Since j was an integer, so is $j + 1$, and our new tree has an odd number of nodes, as required, so $P(k + 1)$ holds.

By the principle of induction, $P(n)$ holds for all $n \in \mathbb{N}$. Since every tree has an (integer) height of 0 or more, every tree is included in some $P()$, so the claim holds for all trees.

(a) What is the bug in the proof? **Solution:**

The proof, in trying to show something about an arbitrary of height $k+1$, builds a **particular** tree of height $k+1$, not an arbitrary one. While the tree built is indeed of height $k+1$ and has an odd number of nodes, it is not an *arbitrary* tree of height $k+1$.

Why is it that in this problem we have to start with $k+1$, but we didn't in the "Walk the Dawgs" problem above? "Walk the Dawgs" is asking you to prove an exists statement ("can split" not "for every possible split..."). When proving an exists, you just say "here's how to do it", you don't need to introduce an arbitrary variable

(b) What should the starting point and target of the IS be (you can't write a full proof, as the claim is false). **Solution:**

"Let T be an arbitrary tree of height $k+1$" should be the first sentence. "T has an odd number of nodes" is the target. Notice that the only difference

6. 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

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

Solution:

For a **Tree** T, let $P(T)$ be "$\mathsf{sum}(T) = \mathsf{sum}(\mathsf{reverse}(T))$". We show $P(T)$ for all **Tree**s T by structural induction.

Base Case: By definition we have $\mathsf{reverse}(\texttt{Nil}) = \texttt{Nil}$. Applying sum to both sides we get $\mathsf{sum}(\texttt{Nil}) = \mathsf{sum}(\mathsf{reverse}(\texttt{Nil}))$, which is exactly $P(\texttt{Nil})$, so the base case holds.

Inductive Hypothesis: Suppose $P(L)$ and $P(R)$ hold for some arbitrary **Tree**s L and R.

Inductive Step: Let x be an arbitrary integer. $\boxed{\text{Goal: Show } P(\texttt{Tree}(x, L, R)) \text{ holds.}}$

We have,

$$\begin{aligned} \mathsf{sum}(\mathsf{reverse}(\texttt{Tree}(x, L, R))) &= \mathsf{sum}(\texttt{Tree}(x, \mathsf{reverse}(R), \mathsf{reverse}(L))) && \text{[Definition of reverse]} \\ &= x + \mathsf{sum}(\mathsf{reverse}(R)) + \mathsf{sum}(\mathsf{reverse}(L)) && \text{[Definition of sum]} \\ &= x + \mathsf{sum}(R) + \mathsf{sum}(L) && \text{[Inductive Hypothesis]} \\ &= x + \mathsf{sum}(L) + \mathsf{sum}(R) && \text{[Commutativity]} \\ &= \mathsf{sum}(\texttt{Tree}(x, L, R)) && \text{[Definition of sum]} \end{aligned}$$

This shows $P(\texttt{Tree}(x, L, R))$.

Conclusion: Therefore, $P(T)$ holds for all **Tree**s T by structural induction.