Section 08: Solutions

1. Strong Induction

Consider the function $a(n)$ defined for $n \geq 1$ recursively as follows.

$$a(1) = 1$$

$$a(2) = 3$$

$$a(n) = 2a(n-1) - a(n-2) \text{ for } n \geq 3$$

Use strong induction to prove that $a(n) = 2n - 1$ for all $n \geq 1$.

Solution:

Let $P(n)$ be "$a(n) = 2n - 1$". We will show that $P(n)$ is true for all $n \geq 1$ by strong induction.

Base Cases $(n = 1, n = 2)$:
$(n = 1)$
$a(1) = 1 = 2 \cdot 1 - 1$

$(n = 2)$
$a(2) = 3 = 2 \cdot 2 - 1$

So, $P(1)$ and $P(2)$ hold.

Inductive Hypothesis:
Suppose that $P(j)$ is true for all integers $1 \leq j \leq k$ for some arbitrary $k \geq 2$.

Inductive Step:
We will show $P(k+1)$ holds.

$$\begin{aligned}
a(k+1) &= 2a(k) - a(k-1) && \text{[Definition of } a\text{]} \\
&= 2(2k-1) - (2(k-1) - 1) && \text{[Inductive Hypothesis]} \\
&= 2k + 1 && \text{[Algebra]} \\
&= 2(k+1) - 1 && \text{[Algebra]}
\end{aligned}$$

So, $P(k+1)$ holds.

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

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

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

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

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.

3. Reversing a Binary Tree

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

Basis Step $\texttt{Nil}$ 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.

4. 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.

Solution:

Basis: $\varepsilon \in S$.
Recursive Step: If $x \in S$, then $x00, x01, x10, x11 \in S$.
Exclusion Rule: Each element of S is obtained from the basis and a finite number of applications of the recursive step.

"Brief " Justification: We will show that $x \in S$ iff x has even length (i.e.,|x|= 2n for some $n \in \mathbb{N}$). (Note: "brief" is in quotes here. Try to write shorter explanations in your homework assignment when possible!)

Suppose $x \in S$. If x is the empty string, then it has length 0, which is even. Otherwise, x is built up from the empty string by repeated application of the recursive step, so it is of the form $x_1 x_2 ... x_n$, where each $x_i \in \{00, 01, 10, 11\}$. In that case, we can see that $|x| = |x_1| + |x_2| + \cdots + |x_n| = 2n$, which is even. Now, suppose that x has even length. If it's length is zero, then it is the empty string, which is in S. Otherwise, it has length 2n for some $n > 0$, and we can write x in the form $x_1 x_2 ... x_n$, where each $x_i \in \{00, 01, 10, 11\}$ has length 2. Hence, we can see that x can be built up from the empty string by applying the recursive step with x_1, then x_2, and so on up to x_n, which shows that $x \in S$.

(b) Binary strings not containing 10.

Solution:

If the string does not contain 10, then the first 1 in the string can only be followed by more 1s. Hence, it must be of the form $0^m 1^n$ for some $m, n \in \mathbb{N}$.

Basis: $\varepsilon \in S$.

Recursive Step: If $x \in S$, then $0x \in S$ and $x1 \in S$.

Exclusion Rule: Each element of S is obtained from the basis and a finite number of applications of the recursive step.

Brief Justification: The empty string satisfies the property, and the recursive step cannot place a 0 after a 1 since it only adds 0s on the left. Hence, every string in S satisfies the property.

In the other direction, from our discussion above, any string of this form can be written as $y = 0^m 1^n$ for some $m, n \in \mathbb{N}$. We can build up the string y from the empty string by applying the rule $x \to 0x$ m times and then applying the rule $x \to x1$ n times. This shows that the string y is in S.

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

Solution:

These must be of the form $0^m 1^n$ for some $m, n \in \mathbb{N}$ with $m \leq n$. We can ensure that by pairing up the 0s with 1s as they are added:

Basis: $\varepsilon \in S$.

Recursive Step: If $x \in S$, then $0x1 \in S$ and $x1 \in S$.

Exclusion Rule: Each element of S is obtained from the basis and a finite number of applications of the recursive step.

Brief Justification: As in the previous part, we cannot add a 0 after a 1 because we only add 0s at the front. And since every 0 comes with a 1, we always have at least as many 1s as 0s.

In the other direction, from our discussion above, any string of this form can be written as xy, where $x = 0^m 1^m$ and $y = 1^{n-m}$, since $n \geq m$. We can build up the string x from the empty string by applying the rule $x \to 0x1$ m times and then produce the string xy by applying the rule $x \to x1$ $n-m$ times, which shows that the string is in S.

(d) Binary strings containing at most two 0s and at most two 1s.

Solution:

This is the set of all binary strings of length at most 4 *except* for these:

$$000, 1000, 0100, 0010, 0001, 0000, 111, 0111, 1011, 1101, 1110, 1111$$

Since this is a **finite set**, we can define it recursively using only basis elements and no recursive step.

5. 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:

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