CSE 311: Foundations of Computing I

Section 7: Strong Induction and Recursive Sets Solutions

1. Cantelli's Rabbits

Xavier Cantelli owns some rabbits. The number of rabbits he has in a given year is described by the function f:

$$\begin{aligned} f(0) &= 0 \\ f(1) &= 1 \\ f(n) &= 2f(n-1) - f(n-2) \qquad \text{for } n \geq 2 \end{aligned}$$

Determine, with proof, the number, $f(n)$, of rabbits that Cantelli owns in year n.

Solution:

Let $\mathsf{P}(n)$ be "$f(n) = n$". We prove that $\mathsf{P}(n)$ is true for all $n \in \mathbb{N}$ by strong induction on n.

Base Cases ($n = 0, 1$): $f(0) = 0$ by definition, so $\mathsf{P}(0)$ holds, and $f(1) = 1$, so $\mathsf{P}(1)$ holds.

Induction Hypothesis: Assume that for some arbitrary integer $k \geq 1$, $\mathsf{P}(j)$ holds for all $0 \leq j \leq k$. That is, for each number in this range, we have $f(j) = j$.

Induction Step: We show $\mathsf{P}(k+1)$, i.e. that $f(k+1) = k+1$.

Since $k + 1 \geq 2$, we have

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

which is $\mathsf{P}(k+1)$.

Therefore, $\mathsf{P}(n)$ is true for all $n \in \mathbb{N}$.

2. Structural Induction

(a) Consider the following recursive definition of strings Σ^* over the alphabet Σ.

Basis Step: ε 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 $\mathsf{leaves}(T) \geq \mathsf{size}(T)/2 + 1/2$. We prove P for all trees T by structural induction.

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

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

Induction Step: We prove that $\mathsf{P}(\texttt{Tree}(\bullet, L, R))$ holds.

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

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

Thus, the $\mathsf{P}(T)$ holds for all trees T.

3. 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 $x_1 x_2 \cdots x_n$, 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 $x_1 x_2 \cdots x_n$, 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 as a substring and having at least as many 1s as 0s.

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 $m, n \in \mathbb{N}$. The second condition says that we have $m \leq n$.

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: The empty string satisfies the property, and the recursive step cannot place a 0 after a 1 since it only adds 1s on the right. 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 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 $x \mapsto 0x1$ m times and then produce the string xy by applying the rule $x \mapsto x1$ $n - m$ times, which shows that the string is in S.

4. 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.)