Section 07: Solutions

1. Recursively Defined Sets of Strings

For each of the following, write a recursive definition of the sets satisfying the following properties. You do not need to mention the exclusion rule. Briefly justify that your solution is correct.

(a) Binary strings of even length.

Solution:

Basis Step: $\varepsilon \in S$.
Recursive Step: If $x \in S$, then $x00, x01, x10, x11 \in S$.

"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 $x_1x_2...x_n$, 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 for the other direction. Suppose that x has even length. If its 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 $x_1x_2...x_n$, 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. So, it must be of the form 0^m1^n for some $m, n \in \mathbb{N}$.

Basis Step: $\varepsilon \in S$.

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

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 (i.e., every string that can be generated) satisfies the property.

In the other direction, from our discussion above, any string of this form can be written as $y = 0^m1^n$ 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^m1^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 Step: $\varepsilon \in S$.

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

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. So 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 $x = 0^m1^m$ 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.

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.

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 $\mathsf{P}(Z)$ holds for some arbitrary string Z.

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

Let c be an arbitrary character. We have

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

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

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 $\mathsf{P}(T)$ be "$\mathsf{leaves}(T) \geq \mathsf{size}(T)/2 + 1/2$." We prove $\mathsf{P}(T)$ holds for all trees T by structural induction.

Base Case (T = •): By definition of leaves, $\mathsf{leaves}(\bullet) = 1$ and by definition of size, $\mathsf{size}(\bullet) = 1$. 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: Goal: Show that $\mathsf{P}(\texttt{Tree}(\bullet, L, R))$ holds.

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

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

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

3. 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 **Trees** T

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

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. Goal: Show $P(\texttt{Tree}(x, L, R))$ 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. One-to-One and Onto

Determine if the following functions are (1) one-to-one and (2) onto. For each claim: provide a proof if true, otherwise give a counterexample (you must also explain why the counterexample works).

(a) $f : \mathbb{N} \to \mathbb{N}$, $f(x) = x^2$ **Solution:**

One-to-One (injective): Let $x, y \in \mathbb{N}$ be arbitrary. Suppose $f(x) = f(y)$. Then $x^2 = y^2$, so either $x = y$ or $x = -y$. Since $x, y \in \mathbb{N}$, either x and y are both 0 or they're both positive integers.

- If $x = 0$ then $y = 0 = -y$, and clearly $x = y$.
- If both are positive, then $x = -y$ is impossible since x is positive and $-y$ is negative. So we must have $x = y$.

In both cases, we have $x = y$. Since x and y were arbitrary, f is one-to-one.

Not onto (not surjective): $5 \in \mathbb{N}$, our codomain, but there is no natural number x such that $f(x) = x^2 = 5$.

(b) $f : \mathbb{R} \to \mathbb{R}$, $f(x) = x^2$ **Solution:**

Not one-to-one (not injective): $f(-2) = 4 = f(2)$**, but** $-2 \neq 2$**.**

Not onto (not surjective): $-5 \in \mathbb{R}$, our codomain, but there is no real number x such that $f(x) = x^2 = -5$ since all real numbers are non-negative when squared.

(c) $f : \mathbb{R}^+ \to \mathbb{R}^+$, $f(x) = x^2$, where $\mathbb{R}^+ = \{x : x \in \mathbb{R} \wedge x \geq 0\}$, i.e., the set of non-negative real numbers. **Solution:**

One-to-one (injective): Let $x, y \in \mathbb{R}^+$ be arbitrary. Suppose $f(x) = f(y)$. Then $x^2 = y^2$, so either $x = y$ or $x = -y$. Since $x, y \in \mathbb{R}^+$, either x and y are both 0 or they're both positive real numbers.

- If $x = 0$ then $y = 0 = -y$, and clearly $x = y$.
- If both are positive, then $x = -y$ is impossible since x is positive and $-y$ is negative. So we must have $x = y$.

In both cases, we have $x = y$. Since x and y were arbitrary, f is one-to-one.

Onto (surjective): Let $x \in \mathbb{R}^+$ be arbitrary. Since x is a positive real number, $\sqrt{x}$ is also a positive real number (i.e., is in our domain). Then $f(\sqrt{(}x)) = (\sqrt{(}x))^2 = x$. Therefore, x is the image of some input under f. Since x was arbitrary, f is surjective.

Notice that the domain and co-domain matter! You have to know both to tell whether the function is one-to-one or onto.

5. A Bijection Proof

Let A be the set of negative integers, i.e., $A = \{-1, -2, -3, ...\}$; let B be the set of integers at least 10, i.e., $B = \{10, 11, 12, 13, ...\}$ Show that $f : A \to B$ defined by $f(x) = |x| + 9$ is a bijection.

You may use these facts:

- for negative numbers x, y: $|x| = |y| \to x = y$
- for negative numbers $|x| = -x$

that **Solution:**

One-to-One: Let x, y be arbitrary elements of A such that $f(x) = f(y)$. By definitions of f, $|x| + 9 = |y| + 9$. Cancelling the 9s, we have $|x| = |y|$. By the fact above, this means that $x = y$. Since x, y were arbitrary, f is one-to-one.

Onto: Let y be an arbitrary element in B. Note that y is a positive integer and $y \geq 10$. Consider the value $x = -y + 9$. Observe that x is negative (since y is positive but greater than 9, negating it will give a negative greater in absolute value than 9, leaving a negative result). Since x is negative, when we calculate $|x| + 9$ we get $|x| + 9 = | - y + 9| + 9 = -(-y + 9) + 9 = y$, as required. Further note that the x we chose is in the set A: we already showed it was negative, and since integers are closed under multiplication and addition, x is an integer. Thus x is a value which gives $f(x) = y$; since y was arbitrary, f is onto.

Since f is both one-to-one and onto, it is a bijection.

6. 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 (again, there should be no leading zeroes.).

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. If k is 1, then y just needs to match any binary string with at least one 1. If k is greater than 1, we can treat the extra 1's at the start of the string as being part of y. This means y still only needs to match binary strings with at least one 1. We are essentially always treating k as 1, because this makes it much easier to match the rest of the string.

Since y just needs to have one 1, we can write y as $(0 \cup 1)^*1(0 \cup 1)*$. Then, we can add our leading 1 to get the final solution.

7. CFGs

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

(a) All binary strings that start with 11.
Solution:

$$\begin{aligned}\mathbf{S} &\to 11\mathbf{T} \\ \mathbf{T} &\to 1\mathbf{T} \mid 0\mathbf{T} \mid \varepsilon\end{aligned}$$

(b) All binary strings that contain at most one 1.
Solution:

$$\begin{aligned}\mathbf{S} &\to \mathbf{ABA} \\ \mathbf{A} &\to 0\mathbf{A} \mid \varepsilon \\ \mathbf{B} &\to 1 \mid \varepsilon\end{aligned}$$

(c) All strings over 0, 1, 2 with the same number of 1s and 0s and exactly one 2.
Hint: First try writing a grammar that just produces binary strings with the same number of 1s and 0s (i.e., doesn't have the "exactly one 2 requirement".
Solution:

$$\mathbf{S} \rightarrow \mathbf{T}2\mathbf{T} \mid \mathbf{ST} \mid \mathbf{TS} \mid 0\mathbf{S}1 \mid 1\mathbf{S}0$$
$$\mathbf{T} \rightarrow \mathbf{TT} \mid 0\mathbf{T}1 \mid 1\mathbf{T}0 \mid \varepsilon$$

T produces binary strings with the same number of 1s and 0s. The **TT** part allows us to concatenate strings together, meaning we can have a string like 1001.

S has multiple components. The **T**2**T** part adds a 2 somewhere in the string. On both sides of the 2 are strings with the same number of 1s and 0s, so the entire string has the same number of 1s and 0s with exactly one 2. The $0\mathbf{S}1$ and $1\mathbf{S}0$ components allow for the 2 to be the middle of the string. The strings on the left and right side of the 2 don't need to be balanced, but the entire string is (i.e. 0002111). Finally, the 2 might not be in the middle of the string (for example, 01120). To account for this, the **ST** and **TS** parts can shift the 2 to the left or right side of a balanced string.