# 3. Recurrence

**3.1. Recursive Definitions.** To construct a **recursively defined function:**

1. Initial Condition(s) (or basis): Prescribe initial value(s) of the function.
2. Recursion: Use a fixed procedure (rule) to compute the value of the function at the integer $n+1$ using one or more values of the function for integers $\leq n$.

To construct a **recursively defined set:**

1. Initial Condition(s) (or basis): Prescribe one or more elements of the set.
2. Recursion: Give a rule for generating elements of the set in terms of previously prescribed elements.

Discussion

In computer programming evaluating a function or executing a procedure is often accomplished by using a **recursion**. A recursive process is one in which one or more initial stages of the process are specified, and the $n$th stage of the process is defined in terms of previous stages, using some fixed procedure. In a computer program this is usually accomplished by using a subroutine, such as a "For" loop, in which the same procedure is repeated a specified number of times; that is, the procedure *calls itself.*

EXAMPLE 3.1.1. *The function $f(n) = 2^n$, where $n$ is a natural number, can be defined recursively as follows:*

*1. Initial Condition:* $f(0) = 1$,

*2. Recursion:* $f(n+1) = 2 \cdot f(n)$, *for* $n \geq 0$.

*For any particular $n$, this procedure could be programmed by first initializing $F = 1$, and then executing a loop "For $i = 1$ to $n$, $2 * F = F$."*

Here is how the definition gives us the first few powers of 2:

$2^1 = 2^{0+1} = 2^0 \cdot 2 = 2$

$2^2 = 2^{1+1} = 2^1 \cdot 2 = 2 \cdot 2 = 4$

$2^3 = 2^{2+1} = 2^2 \cdot 2 = 4 \cdot 2 = 8$

### 3.2. Recursive Definition of the Function $f(n) = n!$.

EXAMPLE 3.2.1. *The factorial function $f(n) = n!$ is defined recursively as follows:*

1. *Initial Condition:* $f(0) = 1$
2. *Recursion:* $f(n+1) = (n+1)f(n)$

Discussion

Starting with the initial condition, $f(0) = 1$, the recurrence relation, $f(n+1) = (n+1)f(n)$, tells us how to build new values of $f$ from old values. For example,

$1! = f(1) = 1 \cdot f(0) = 1,$

$2! = f(2) = 2 \cdot f(1) = 2,$

$3! = f(3) = 3 \cdot f(2) = 6,$ etc.

When a function $f(n)$, such as the ones in the previous examples, is defined recursively, the equation giving $f(n+1)$ in terms of previous values of $f$ is called a **recurrence relation**.

### 3.3. Recursive Definition of the Natural Numbers.

DEFINITION 3.3.1. *The set of natural numbers may be defined recursively as follows.*

1. *Initial Condition:* $0 \in \mathbb{N}$
2. *Recursion: If $n \in \mathbb{N}$, then $n+1 \in \mathbb{N}$.*

Discussion

There are a number of ways of defining the set $\mathbb{N}$ of natural numbers recursively. The simplest definition is given above. Here is another recursive definition for $\mathbb{N}$.

EXAMPLE 3.3.1. *Suppose the set $S$ is defined recursively as follows:*

1. *Initial Condition:* $0, 1 \in S$,
2. *Recursion: If $x, y \in S$, then $x + y \in S$.*

*Then* $S = \mathbb{N}$.

*Notice that, in the recursive step,* $x$ *and* $y$ *don't have to represent different numbers. Thus, having* $x = y = 1 \in S$*, we get* $1 + 1 = 2 \in S$*. Then we get* $1 + 2 = 3 \in S$*. And so on.*

It should be noted that there is an *extremal clause* in recursively defined sets. If you cannot build a given element in a finite number of applications of the recursion then it is not in the set built from the recursive definition. To prove an element is in a recursively defined set, you must show that element can be built in a finite number of steps.

EXAMPLE 3.3.2. *Prove that the set* $S$ *recursively in Example 3.3.1 is equal to the set* $\mathbb{N}$ *of natural numbers.*

PROOF. We will show that $S = \mathbb{N}$ by showing separately that $S \subseteq \mathbb{N}$ and $\mathbb{N} \subseteq S$.

1. First we show $\mathbb{N} \subseteq S$. Prove by induction that $n \in S$ for every natural number $n \geq 0$.
   (a) Basis Step. $0, 1 \in S$ by definition.
   (b) Induction Step. Suppose $n \in S$, for some $n \geq 1$. Then, by the recursion step, $n \in S$ and $1 \in S$ imply $n + 1 \in S$.

   Thus, by the first principle of mathematical induction, $n \in S$ for every natural number $n \geq 0$.
2. Now we show $S \subseteq \mathbb{N}$. This time we apply the second principle of mathematical induction on $n$ to show that if $s \in S$ is produced by applying $n$ steps (1 initial condition and $n - 1$ recursive steps), then $s \in \mathbb{N}$.
   (a) Basis Step. After one step the only elements produced are 0 and 1, each of which is in $\mathbb{N}$.
   (b) Induction Step. Suppose $n \geq 1$ and assume any element in $S$ produced by applying $n$ or fewer steps is also an element of $\mathbb{N}$. Suppose $s \in S$ is produced by applying $n + 1$ steps. Since $n + 1 \geq 2$, there must be two elements $x$ and $y$ in $S$, such that $s = x + y$. Both $x$ and $y$ must have been produced by applying *fewer* than $n + 1$ steps, since $s$ is produced by applying $n + 1$ steps, and we use one step to obtain $s$ from $x$ and $y$. By the induction hypothesis both $x$ and $y$ are elements of $\mathbb{N}$. Since the sum of two natural numbers is again a natural number we have $s \in \mathbb{N}$.

□

**3.4. Proving Assertions About Recursively Defined Objects.** Assertions about recursively defined objects are usually proved by mathematical induction. Here are three useful versions of induction. In particular, note the third version which we introduce here.

Version 1. *Second Principle of Induction*

a. Basis Step: Prove the assertion for the initial conditions. (The assertion may have to be verified for more than one particular value.)

b. Induction Step: Assume the assertion is true for integers $\leq n$, and use the recurrence relation to prove the assertion for the integer $n+1$.

Version 2. a. Basis Step: Prove the assertion for the initial conditions. (The assertion may have to be verified for more than one particular value.)

b. Induction Step: Assume the assertion is true when the recursive definition has been applied less than or equal to n times for some integer $n \geq 0$, and use the recurrence relation to prove the assertion when the recursive definition is applied $n+1$ times.

Version 3. *Generalized or Structural Principle of Induction*: Use to prove an assertion about a set $S$ defined recursively by using a set $X$ given in the basis and a set of rules using $s_1, s_2, \ldots, s_k \in S$ for producing new members in the recursive set.

a. Basis Step: Prove the assertion for every $s \in X$.

b. Induction Step: Let $s_1, s_2, \ldots, s_k \in S$ be arbitrary and assume the assertion for these elements (this is the induction hypothesis). Prove all elements of $S$ produced using the recursive definition and $s_1, s_2, \ldots, s_k$ satisfies the assertion.

Discussion

EXAMPLE 3.4.1. *Let $S$, a subset of $\mathbb{N} \times \mathbb{N}$, be defined recursively by*

1. *Initial Condition:* $(0,0) \in S$
2. *Recursion: If $(m,n) \in S$, then $(m+2, n+3) \in S$.*

*Prove that if $(m,n) \in S$, then $m+n$ is a multiple of 5.*

PROOF. We use the Generalized Principle of Induction.

1. Basis Step: Show the statement for $(0,0)$: $0+0$ is a multiple of 5. Since $0 = 0 \cdot 5$ this is clear.
2. Inductive Step: Let $(m,n) \in S$ and assume $m+n$ is a multiple of 5. Show the statement is true for $(m+2, n+3)$. In other words, show $(m+2)+(n+3)$ is a multiple of 5.

$$(m+2)+(n+3) = (m+n)+5$$

We know $m+n$ is a multiple of 5 and clearly 5 is a multiple of 5, so the sum must also be a multiple of 5. This proves the induction step.

Therefore, by the first principle of mathematical induction if $(m, n) \in S$, then $m + n$ is a multiple of 5. □

EXERCISE 3.4.1. *Suppose the set $S$ is defined recursively as follows:*

1. $1 \in S$,
2. *If $x \in S$, then $2 \cdot x \in S$.*

*Prove that $S = \{2^n | n \geq 0\}$.*

EXERCISE 3.4.2. *Suppose the set $S$ is defined recursively as follows:*

1. $0, 1 \in S$,
2. *If $x, y \in S$, then $x \cdot y \in S$.*

*What are the elements of $S$? Prove that your answer is correct.*

EXAMPLE 3.4.2. *Suppose $f : \mathbb{N} \to \mathbb{R}$ is define recursively by*

1. *Initial Condition:* $f(0) = 0$
2. *Recursion:* $f(n + 1) = f(n) + (n + 1)$, *for* $n \geq 0$.

*Then* $f(n) = \dfrac{n(n+1)}{2}$ *for all* $n \geq 0$

PROOF. 1. Basis Step ($n = 0$): $f(0) = 0$, by definition. On the other hand $\dfrac{0(0+1)}{2} = 0$. Thus, $f(0) = \dfrac{0(0+1)}{2}$.

2. Inductive Step: Suppose $f(n) = \dfrac{n(n+1)}{2}$ for some $n \geq 0$. We must prove $f(n+1) = \dfrac{(n+1)((n+1)+1)}{2}$.

$$\begin{aligned} f(n+1) &= f(n) + (n+1) && \text{(recurrence relation)} \\ &= \frac{n(n+1)}{2} + (n+1) && \text{(by the induction hypothesis)} \\ &= (n+1)\left[\frac{n}{2} + 1\right] \\ &= (n+1)\left[\frac{n+2}{2}\right] \\ &= \frac{(n+1)((n+1)+1)}{2} \end{aligned}$$

Therefore, by the first principle of mathematical induction $f(n) = \dfrac{n(n+1)}{2}$ for all $n \geq 0$. □

EXERCISE 3.4.3. *Suppose $f : \mathbb{N} \to \mathbb{R}$ is define recursively as follows:*

1. $f(0) = 0$,
2. $f(n+1) = f(n) + (2n+1)$, *for* $n \geq 0$.

*Prove that $f(n) = n^2$ for all $n \geq 0$.*

## 3.5. Definition of $f^n$.

DEFINITION 3.5.1. *Let $f : A \to A$ be a function. Then we define $f^n$ recursively as follows*

1. *Initial Condition:* $f^1 = f$
2. *Recursion:* $f^{n+1} = f \circ f^n$, *for* $n \geq 1$.

Discussion

EXAMPLE 3.5.1. *Prove that if $f$ is injective, then $f^n$ is injective for $n \geq 1$.*

PROOF. Assume $f$ is injective.

1. Basis Step: $f^1 = f$ is injective by our assumption.
2. Inductive Step: Let $n \geq 1$ and assume $f^n$ is injective. Prove $f^{n+1}$ is injective.
   Recall that to prove $f^{n+1}$ is injective we must show $\forall a, b \in A[(f^{n+1}(a) = f^{n+1}(b)) \to (a = b)]$
   Assume $a, b \in A$ and $f^{n+1}(a) = f^{n+1}(b)$.

$$\begin{aligned}
f^{n+1}(a) &= f^{n+1}(b) && \text{(recurrence relation)}\\
(f \circ f^n)(a) &= (f \circ f^n)(b) && \text{(recurrence relation)}\\
f(f^n(a)) &= f(f^n(b)) && \text{(by the definition of composition)}\\
f^n(a) &= f^n(b) && \text{(since f is injective)}\\
a &= b && \text{(by the induction hypothesis, } f^n \text{ is injective)}
\end{aligned}$$

Therefore, by the first principle of mathematical induction $f^n$ is injective for all positive integers.

□

EXERCISE 3.5.1. *Prove that if $f$ is surjective that $f^n$ is surjective.*

### 3.6. Example 3.6.1.

EXAMPLE 3.6.1. *Given a real number $a \neq 0$, define $a^n$ for all natural numbers, $n$, inductively by*

1. *Initial Condition:* $a^0 = 1$
2. *Recursion:* $a^{(n+1)} = a^n a$

THEOREM 3.6.1. $\forall m \forall n[a^m a^n = a^{m+n}]$ *where $m, n$ are natural numbers.*

PROOF. Proof that $(\forall m)(\forall n)[a^m a^n = a^{m+n}]$, where $m, n$ are natural numbers. We accomplish this by assuming $m$ is an arbitrary natural number and proving $\forall n[a^m a^n = a^{m+n}]$ by induction on $n$.

1. Basis Step ($n = 0$): Show $a^m a^0 = a^{m+0}$.
   This follows directly from the initial condition of the definition: $a^0 = 1$, therefore $a^m a^0 = a^m(1) = a^m = a^{m+0}$.
2. Induction Step:
   Induction hypothesis: Let $n$ be a natural number and assume $a^m a^n = a^{n+m}$. Prove $a^m a^{n+1} = a^{m+(n+1)}$.

$$\begin{aligned} a^m a^{n+1} &= a^m a^n a && \text{by the recursive part of the definition: } a^{n+1} = a^n a \\ &= a^{m+n} a && \text{by the induction hypothesis} \\ &= a^{(m+n)+1} = a^{(m+(n+1))} && \text{by the recursive part of the definition} \end{aligned}$$

   By induction, $\forall n[a^m a^n = a^{m+n}]$.

Since $m$ was an arbitrary natural number the statement is true for all natural numbers $m$. □

Discussion

Here we see a recursive definition for the function $f(n) = a^n$, where $n$ is a natural number and $a$ is an arbitrary nonzero real number followed by a proof of one of the laws of exponents, $a^{m+n} = a^m a^n$. This proof uses both mathematical induction and Universal Generalization. We fix $m$ as some arbitrary natural number and then proceed to use induction on $n$. We do not need to use induction on both $m$ and $n$ simultaneously. When there are two different variables, this is a standard strategy to try. There are circumstances, however, when this strategy doesn't work so that you

would need to use induction on *both* of the variables (*double induction*). We will not encounter these kinds of problems in this course, however.

## 3.7. Fibonacci Sequence.

DEFINITION 3.7.1. *The Fibonacci sequence may be defined recursively as follows:*

1. *Initial Conditions:* $F(0) = 0, F(1) = 1$

2. *Recursion:* $F(n+1) = F(n) + F(n-1)$ *for* $n \geq 1$

Discussion

The famous *Fibonacci sequence* is defined here using a recursively defined function. The definition of the Fibonacci sequence requires two initial conditions. There are no limits on the number of initial conditions on a recursively defined object – only that there be a fixed finite number in each instance.

EXAMPLE 3.7.1. *Suppose* $F(n)$, $n \geq 0$, *denotes the Fibonacci sequence. Prove that* $1 < \dfrac{F(n+1)}{F(n)} < 2$ *for all* $n \geq 3$.

PROOF. Let $R(n) = \dfrac{F(n+1)}{F(n)}$ for $n \geq 1$. We will prove by induction that

$$1 < R(n) < 2$$

for all $n \geq 3$.

1. Basis Step ($n = 3$): $R(3) = \dfrac{F(4)}{F(3)} = \dfrac{3}{2}$, and $1 < \frac{3}{2} < 2$.

2. Induction Step: Suppose $1 < R(n) < 2$ for some $n \geq 3$. We need to prove $1 < R(n+1) < 2$.

$$\begin{aligned} R(n+1) &= \frac{F(n+2)}{F(n+1)} \\ &= \frac{F(n+1)+F(n)}{F(n+1)} \quad \text{by the recursive definition of } F(n) \\ &= 1 + \frac{F(n)}{F(n+1)} \qquad \text{or} \\ R(n+1) &= 1 + \frac{1}{R(n)} \end{aligned}$$

By the inductive hypothesis $1 < R(n) < 2$; and so

$$1 > \frac{1}{R(n)} > \frac{1}{2}.$$

Thus,

$$2 > 1 + \frac{1}{R(n)} > \frac{3}{2} > 1,$$

or

$$1 < 1 + \frac{1}{R(n)} < 2.$$

Substituting from above, we have

$$1 < R(n+1) < 2.$$

By the first principle of mathematical induction, $1 < \dfrac{F(n+1)}{F(n)} < 2$ for all $n \geq 3$.

□

EXAMPLE 3.7.2. *(calculus required) Prove that* $\lim\limits_{n\to\infty} \dfrac{F(n+1)}{F(n)} = \dfrac{1+\sqrt{5}}{2}$, *if the limit exists.*

PROOF. Let $R(n) = \dfrac{F(n+1)}{F(n)}$ as in Example 3.7.1. Then, from the induction step in Example 3.7.1, we see that $R(n+1) = 1 + \dfrac{1}{R(n)}$. Assume $\lim\limits_{n\to\infty} \dfrac{F(n+1)}{F(n)}$ exists and let $L = \lim\limits_{n\to\infty} \dfrac{F(n+1)}{F(n)} = \lim\limits_{n\to\infty} R(n)$. Notice that $\lim\limits_{n\to\infty} R(n) = \lim\limits_{n\to\infty} R(n+1)$. Therefore, $L = \lim\limits_{n\to\infty} \big(1 + \dfrac{1}{R(n)}\big) = 1 + \dfrac{1}{L}$.

Since $L$ is a real number, the equation

$$L = 1 + \frac{1}{L}$$

is equivalent to

$$L^2 - L - 1 = 0.$$

By the quadratic formula

$$L = \frac{1 \pm \sqrt{5}}{2}.$$

Since $L > 0$ and since $\sqrt{5} > 1$,

$$L = \frac{1 + \sqrt{5}}{2}.$$

□

EXERCISE 3.7.1. *Prove that* $F(n) > (\frac{3}{2})^{n-1}$ *for* $n \geq 6$. *[Hint: Show that the statement is true for* $n = 6$ *and 7 (basis step), and then show the induction step works for* $n \geq 7$.*]*

Here is another "Fibonacci-like" sequence.

EXAMPLE 3.7.3. *Suppose* $F(n)$, $n \geq 0$, *is defined recursively as follows:*

1. $F(0) = 1$ *and* $F(1) = 2$,
2. $F(n+1) = F(n) + 2F(n-1)$, *for* $n \geq 1$.

*Prove that* $F(n) = 2^n$ *for all* $n \geq 0$.

PROOF. (Using the *second* principle of mathematical induction)

1. Basis Step ($n = 0$ and $n = 1$): $F(0) = 1 = 2^0$ and $F(1) = 2 = 2^1$.
2. Induction Step: Let $n \geq 1$. Suppose $F(k) = 2^k$ for $0 \leq k \leq n$. Then

$$\begin{aligned} F(n+1) &= F(n) + 2F(n-1) \\ &= 2^n + 2 \cdot 2^{n-1} \\ &= 2^n + 2^n \\ &= 2 \cdot 2^n \\ &= 2^{n+1} \end{aligned}$$

Thus, by the second principle of mathematical induction, $F(n) = 2^n$ for all $n \geq 0$. □

REMARKS 3.7.1. *(1) Here and in the previous exercise we see the slight variation in the basis step from the ones encountered in Module 3.3 Induction; there may be more than one initial condition to verify before proceeding to the induction step.*

*(2) Notice that in this example as well as in some of the other examples and exercises, we have been asked to prove that a function defined recursively is also given by a relatively simple formula. The problem of "solving" recurrence relations for these nice formulas (so-called closed form) is an interesting subject in its own right, but it will not be discussed in this course.*

EXERCISE 3.7.2. *Let $f : \mathbb{N} \to \mathbb{Z}$ be the function defined recursively by*

1. *$f(0) = 1$ and $f(1) = -4$,*
2. *$f(n) = -3f(n-1) + 4f(n-2)$, for $n \geq 2$.*

*Prove $f(n) = (-4)^n$*

EXERCISE 3.7.3. *Let $f : \mathbb{N} \to \mathbb{Z}$ be the function defined recursively by*

1. *$f(0) = 2$ and $f(1) = 7$,*
2. *$f(n) = f(n-1) + 2f(n-2)$, for $n \geq 2$.*

*Prove $f(n) = 3 \cdot 2^n - (-1)^n$*

### 3.8. Strings.

DEFINITION 3.8.1. *Given a finite set of symbols, $\Sigma$, the set of strings, denoted $\Sigma^*$, over $\Sigma$ is defined recursively as follows:*

1. *Initial Condition: The empty string $\lambda$ is in $\Sigma^*$.*
2. *Recursion: If $w$ is a string in $\Sigma^*$ and $a$ is a symbol in $\Sigma$, then $wa$ is in $\Sigma^*$.*

Discussion

The set $\Sigma$ is usually called an **alphabet**, and strings in $\Sigma^*$ are called **words** in the alphabet $\Sigma$. Strings ("words") on a finite alphabet, $\Sigma$, are defined recursively using *right concatenation*. In other words, every string of symbols from $\Sigma$ can be built from a smaller string by applying new letters to the right.

REMARK 3.8.1. *When $a$ is a symbol from an alphabet, we use the notation $a^n$, where $n \in \mathbb{N}$, to represent $a$ concatenated with itself $n$ times. In particular, $a^0$ is understood to represent the empty string, $\lambda$.*

EXERCISE 3.8.1. *Suppose the alphabet $\Sigma$ is the set $\{a, b, c, d\}$. Then $\Sigma^*$ is the set of all words on $\Sigma$. Use right concatenation to build the bit string daabc starting with the empty string, $\lambda$ (use $\lambda a = a$ for any element, $a$, in $\Sigma$).*

EXERCISE 3.8.2. *Now take the same bit string, daabc, and build it using left concatenation. Notice that your steps are not the same; that is, concatenation is not commutative. Regardless, we arrive at the same set of strings, $\Sigma^*$.*

## 3.9. Bit Strings.

EXAMPLE 3.9.1. *The set $S$ of all bit strings with no more than a single 1 can be defined recursively as follows:*

*1. Initial Condition:* $\lambda, 1 \in S$

*2. Recursion: If $w$ is a string in $S$, then so are $0w$ and $w0$.*

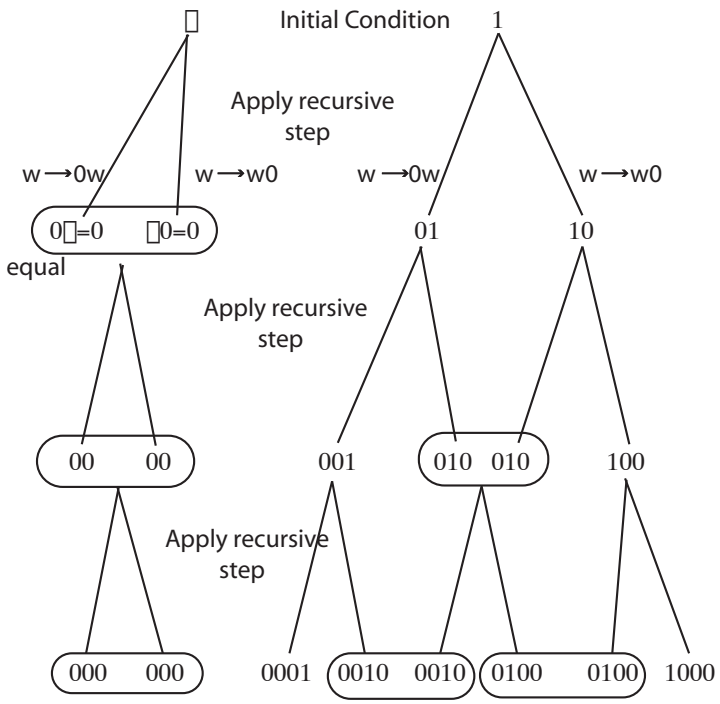
Discussion

In this definition we must start with two objects in the set. Then we can build all the bit strings that have *at most* one "1". We can define various subsets of bit strings using recursively defined sets.

EXAMPLE 3.9.2. *This example is an extension of example 3.9.1. Let the set $S$ be defined recursively by*

**Basis:** $\lambda, 1 \in S$
**Recursion:** *If $w \in S$ then $0w \in S$ and $w0 \in S$.*

*In creating a set recursively, it can help to use a tree diagram to develop more of an intuitive understanding of how this set is built. The following diagram shows how the applying the above definition 4 times gives the elements in the diagram. Some key ideas to keep in mind is that all strings in the tree and all strings that would be in the tree if you kept going are in the set. When a string is repeated, that means there is more than one way to get that element but there is no need to see what is produced from it more than one time.*




*Prove $S$ is the set of all bit strings with no more than one 1.*

PROOF. Let $A$ denote the set of all bit strings with no more than one 1 in the string. Then we need to show $A = S$.

First we will show $A \subseteq S$. Let $a \in A$. Then $a$ is a bit string with either no 1's or it is a bit string with exactly one 1.

Case 1 Suppose $a$ has no 1's. Then $a = 0^n$ where $n$ is some natural number. We can build $a$ using the recursive definition by starting with $\lambda$ and applying the recursive step $n$ times. (If we apply the recursive step 0 times, we get $\lambda$).

Case 2 Suppose $a$ has exactly on 1. Then $a = 0^n 10^m$ for some $n, m \in \mathbb{N}$. We can build $a$ by starting with 1, which is given by the initial condition, and applying the recursive step $(w \in S) \to (0w \in S)$ $n$ times and applying $(w \in S) \to (w0 \in S)$ $m$ times.

This shows we can apply the recursive definition given for $S$ finitely many times to obtain any element of $A$. Therefore $A \subseteq S$.

Now we show $S \subseteq A$ by general induction.

**basis:** The elements given by the basis (initial condition) of the defintion of $S$ are both in $A$ since $\lambda$ has no 1's and 1 has one 1.

**induction step:** Let $x \in S$ and assume $x \in A$. We need to show any elements obtained by appling the recursive step one time will also be in $A$.

Notice we obtain $0x$ and $x0$ when we apply the recursive step one time to $x$. Since $x$ is in $A$ we know the string $x$ has either no ones or a single one. $0x$ and $x0$ do not add any more 1's to the bit string, so they are also in $A$.

Thus by the principle of mathematical induction $\forall x \in S(x \in A)$.

This completes the proof that $S = A$. □

EXAMPLE 3.9.3. *Here's an example of proving a recursively defined set of bit strings is the same as set of bit strings defined in a non-recursive manner:*

*Let $A$ be the set of all bit strings of the form $0^n1^n$, where $n$ can be any natural number. Note that this is the same as $A = \{0^n1^n | n \in \mathbb{N}\} = \{\lambda, 01, 0011, 000111, 00001111, ...\}$ and is slightly different from the set described in Exercise 3.9.4.*

*Now define $B$ by the following recursive definition:*

**Basis:** $\lambda \in B$
**Recursive Step:** *If $w \in B$ then $0w1 \in B$*

*Prove that $A = B$.*

PROOF. First we prove $A \subseteq B$. Let $a \in A$. Then $a = 0^n1^n$ for some $n \in \mathbb{N}$. If we use the recursive definition of $B$ we see $\lambda = 0^01^0$ by the basis step and we claim that if we apply the recursive step $n$ times to $\lambda$ we will build to the element $0^n1^n$. This will demonstrate that we can apply the recursive definition to find $a$ using a finite number of steps. Thus $a \in B$. To prove the claim we use the first principle of induction on the predicate $P(n) =$"If we apply the recursive step $n$ times to $\lambda$ we will build to the element $0^n1^n$."

EXERCISE 3.9.1. *Prove the above claim.*

Now we need to prove $B \subseteq A$. We will do this using generalized induction, which gives us a formal proof of this statement.

**Basis:** Notice the element, $\lambda$, created by the initial step (or basis step) in the definition of $B$ is also an element of $A$ ($\lambda = 0^01^0$).

**Induction Step:** Let $x \in B$ be such that $x \in A$ as well. Show that any element of $B$ obtained from $x$ by applying the recursive step one time is also an element of $A$.

If we apply the recursive step to $x$ one time the only element we get $0x1$. Since $x$ is an element of $A$ we know $x = 0^n1^n$ for some $n \in \mathbb{N}$. So then $0x1 = 0(0^n1^n)1 = 0^{n+1}1^{n+1}$ which we see is also an element of $A$.

Thus by the principle of generalized induction $\forall x \in B(x \in A)$.

This completes that proof that $A = B$. □

EXERCISE 3.9.2. *What kinds of bit strings would we have if the initial condition in Example 3.9.1 is changed to* $1 \in S$ *only? So the definition would be*

1. *Initial Condition:* $1 \in S$,
2. *Recursion: If* $w$ *is a string in* $S$*, then so are* $0w$ *and* $w0$.

EXERCISE 3.9.3. *What kinds of strings do we get from the following recursive definition?*

1. *Initial Conditions:* $\lambda, 1 \in S$,
2. *Recursion: If* $w$ *is a string in* $S$*, then so is* $0w$.

EXERCISE 3.9.4. *Find a recursive definition for the set of bit strings* $T = \{0^r1^s | r, s \in \mathbb{N}\}$.

EXERCISE 3.9.5. *Prove your answer for Exercise 3.9.4 is correct.*

EXERCISE 3.9.6. *Find a recursive definition for the set of all bit strings containing no adjacent 1's. (For example, 1001010 is allowed, but 0011010 is not.)*