2. MATHEMATICAL INDUCTION AND RECURSIVE DEFINITIONS

2.1 Proofs

- A *proof* of a statement is a convincing argument that the statement is true.

- A good proof not only convinces but also explains why the statement is true.

- A typical step in a proof uses general principles of logical reasoning to derive some statement from:

  Assumptions or hypotheses
  Statements that have already been derived
  Other generally accepted facts

- It is common to take shortcuts, assuming that the reader could fill in low-level details.

Direct Proofs

- The statement to be proved usually has the form $p \rightarrow q$. A *direct* proof assumes that the statement p is true and uses this to show q is true.

- **Example 2.1:** The Product of Two Odd Integers Is Odd

  To prove: For any integers a and b, if a and b are odd, then ab is odd.

  Proof: An integer n is odd if there exists an integer x so that $n = 2x + 1$.
  If a and b are any odd integers, then there is an integer x so that $a = 2x + 1$, and there is an integer y so that $b = 2y + 1$.
  Showing that ab is odd involves showing that there exists z such that $ab = 2z + 1$.
  The value of ab is $(2x + 1)(2y + 1) = 4xy + 2x + 2y + 1 = 2(2xy + x + y) + 1$.
  Let $z = 2xy + x + y$. Then $ab = 2z + 1$, so ab is odd.

- This is an example of a *constructive* proof, because it finds a specific value for z, rather than simply showing that z exists.

- Proof techniques for statements that involve quantifiers:

  To prove "There exists x such that $P(x)$": Find a value of x that makes $P(x)$ true.

  To *disprove* "For every x, $P(x)$": Find a value of x for which $P(x)$ is false. (This is called a *proof by counterexample*.)

  To prove "For every x, $P(x)$": Argue that $P(x)$ is true without giving any restrictions on x.

Indirect Proofs

- The alternative to a direct proof is an *indirect proof*.

- The simplest form of indirect proof is a *proof by contrapositive*, using the logical equivalence of $p \rightarrow q$ and $\neg q \rightarrow \neg p$.

- **Example 2.2:** A Proof by Contrapositive

  To prove: For any positive integers *i*, *j*, and *n*, if $i * j = n$, then either $i \leq \sqrt{n}$ or $j \leq \sqrt{n}$.

  Proof: The statement to be proved is equivalent to: For any positive integers *i*, *j*, and *n*, if it is not the case that $i \leq \sqrt{n}$ or $j \leq \sqrt{n}$, then $i * j \neq n$.
  If it is not true that $i \leq \sqrt{n}$ or $j \leq \sqrt{n}$, then $i > \sqrt{n}$ and $j > \sqrt{n}$.
  If *a* and *b* are numbers with $a > b$, and *c* is a number > 0, then $ac > bc$.
  Applying this to the inequality $i > \sqrt{n}$ with $c = j$, we obtain $i * j > \sqrt{n} * j$.
  Since $n > 0$, we know that $\sqrt{n} > 0$, so the same fact can be applied again to the inequality $j > \sqrt{n}$,this time letting $c = \sqrt{n}$, to obtain $j\sqrt{n} > \sqrt{n}\sqrt{n} = n$.
  We now have $i * j > j\sqrt{n} > n$, and it follows that $i * j \neq n$.

- This proof omits a number of details, relying on the reader's knowledge of logic and the properties of real numbers.

- A good proof strikes a balance: leaving out minor details that obscure the main points but containing enough detail to convince the reader.

Proof by Contradiction

- A variation of proof by contrapositive is *proof by contradiction*.

- Proving a statement *p* by contradiction means showing that if it is not true, some contradiction results.

- **Example 2.3:** $\sqrt{2}$ Is Irrational

  A real number *x* is *rational* if there are two integers *m* and *n* so that $x = m / n$. The proof that $\sqrt{2}$ is irrational is one of the most famous examples of proof by contradiction.

  Proof: Suppose for the sake of contradiction that $\sqrt{2}$ is rational.
  Then there are integers m' and n' with $\sqrt{2} = m' / n'$.
  By dividing both m' and n' by all the factors that are common to both, we obtain $\sqrt{2} = m / n$, for some integers *m* and *n* having no common factors.
  Since $m / n = \sqrt{2}$, $m = n\sqrt{2}$.
  Squaring both sides of this equation, we obtain $m^2 = 2n^2$, and therefore m^2 is even (divisible by 2).
  But if m^2 is even, *m* must be even, so $m = 2k$ for some *k*.
  Therefore, $(2k)^2 = 2n^2$.
  Simplifying this and canceling 2 from both sides, we obtain $2k^2 = n^2$. Therefore, n^2 is even.
  Since n^2 is even, *n* must be even, and so $n = 2j$ for some *j*.
  The fact that that *m* and *n* are both divisible by 2 contradicts the previous statement that *m* and *n* have no common factor.
  The assumption that $\sqrt{2}$ is rational therefore leads to a contradiction, and the conclusion is that $\sqrt{2}$ is irrational.

Proof by Cases

- Another useful technique is to divide a proof into separate cases.

- **Example 2.6:** Strings of Length 4 Contain Substrings *yy*

  To prove: Every string *x* in {0, 1}* of length 4 contains a nonnull substring of the form *yy.*

  Proof: Consider the following two cases.
  (1) If *x* contains two consecutive 0's or two consecutive 1's, then the statement is true for a string *y* of length 1.
  (2) Otherwise, any symbol that follows a 0 must be a 1, and vice versa, so *x* must be 0101 or 1010. The statement is therefore true for a string *y* of length 2.

- There is often a choice of cases. In Example 2.6, the two cases could have been divided into subcases. Alternatively, all 16 strings of length 4 could have been considered separately.

- Any division into cases is valid, as long as the cases cover all the possibilities and the proof can be completed for each case.

2.2 The Principle of Mathematical Induction

- It is often necessary to prove that some statement involving a natural number n is true for every sufficiently large value of n.

- Examples:

  $$\sum_{i=1}^{n} i = n(n+1)/2$$

  The number of subsets of $\{1, 2, \ldots, n\}$ is 2^n.

  $n! > 2^n$

  There exist positive integers j and k so that $n = 3j + 7k$.

  Every language with exactly n elements is regular.

  If $x \in \{0, 1\}^*$, $|x| = n$, and $x = 0y1$, then x contains the substring 01.

- The last two statements didn't need to be formulated in terms of n, but doing so allows the use of the proof technique known as mathematical induction.

- **The Principle of Mathematical Induction:** Suppose $P(n)$ is a statement involving an integer n. Then to prove that $P(n)$ is true for every $n \geq n_0$, it is sufficient to show these two things:

  1. $P(n_0)$ is true.
  2. For any $k \geq n_0$, if $P(k)$ is true, then $P(k + 1)$ is true.

Proof By Induction

- A *proof by induction* is an application of the principle of mathematical induction.

- The two parts of such a proof are called the *basis step* and the *induction step*.

- In the induction step, we assume that k is a number $\geq n_0$ and that the statement $P(n)$ is true in the case $n = k$; this assumption is called the *induction hypothesis*.

- **Example 2.7:** Let $P(n)$ be the statement

  $1 + 2 + 3 + \ldots + n = n(n + 1)/2$

  To show that $P(n)$ is true for every $n \geq 0$.

  Basis step. $P(0)$ is the statement $0 = 0(0 + 1)/2$, which is obviously true.

  Induction hypothesis.

  $k \geq 0$ and $1 + 2 + 3 + \ldots + k = k(k + 1)/2$

  Statement to be shown in induction step.

  $1 + 2 + 3 + \ldots + (k + 1) = (k + 1)((k + 1) + 1)/2$

  Proof of induction step.

  $$\begin{aligned} 1 + 2 + 3 + \ldots + (k + 1) &= (1 + 2 + \ldots + k) + (k + 1) \\ &= k(k + 1)/2 + (k + 1) \quad \text{(by the induction hypothesis)} \\ &= (k + 1)(k/2 + 1) \\ &= (k + 1)(k + 2)/2 \\ &= (k + 1)((k + 1) + 1)/2 \end{aligned}$$

Proof By Induction (Continued)

- An induction proof should always be explicit about the following:

  The general statement involving n that is to be proved.
  The statement to which it reduces in the basis step (the general statement, but with n_0 substituted for n).
  The induction hypothesis (the general statement, with k substituted for n, and preceded by "$k \geq n_0$, and").
  The statement to be shown in the induction step (with $k + 1$ substituted for n).
  The point during the induction step at which the induction hypothesis is used.

Proof By Induction (Continued)

- **Example 2.8:** Strings of the Form 0*y*l Must Contain the Substring 01

  To prove: For any $x \in \{0, 1\}^*$, if x begins with 0 and ends with 1 (i.e., $x = 0y1$ for some string y), then x must contain the substring 01.

  Proof: In order to use mathematical induction, the statement to be proved must be reformulated in terms of an integer n.
  Let $P(n)$ be the statement: If $|x| = n$ and $x = 0y1$ for some string $y \in \{0, 1\}^*$, then x contains the substring 01.
  The goal is now to prove that for every $n \geq 2$, $P(n)$ is true.

  Basis step. $P(2)$ says that if $|x| = 2$ and $x = 0y1$ for some $y \in \{0, 1\}^*$, then x contains the substring 01. $P(2)$ is true, because if $|x| = 2$ and $x = 0y1$ for some y, then $x = 01$.

  Induction hypothesis. $k \geq 2$ and $P(k)$; in other words, if $|x| = k$ and $x = 0y1$ for some $y \in \{0, 1\}^*$, then x contains the substring 01.

  Statement to be shown in induction step. $P(k + 1)$; that is, if $|x| = k + 1$ and $x = 0y1$ for some $y \in \{0, 1\}^*$, then x contains the substring 01.

  Proof of induction step. Since $|x| = k + 1$ and $x = 0y1$, $|y1| = k$. If y begins with 1, then x begins with the substring 01. If y begins with 0, then $y1$ begins with 0 and ends with 1; by the induction hypothesis, y contains the substring 01, and therefore x does also.

- The same result can be shown without explicit use of induction:

  If $x = 0y1$ for some string $y \in \{0, 1\}^*$, then x must contain at least one 1.
  The *first* 1 in x cannot occur at the beginning, since x starts with 0.
  Therefore, the first 1 must be immediately preceded by a 0, which means that x contains the substring 01.

- Although this proof does not mention induction, it relies on a fact about natural numbers (every nonempty subset has a smallest element) that is equivalent to the principle of mathematical induction.

The Minimal Counterexample Principle

- The principle of mathematical induction sometimes appears in a disguised form, called the *minimal counterexample principle*.

- **Example 2.10:** A Proof Using the Minimal Counterexample Principle

  To prove: For every integer $n \geq 0$, $5^n - 2^n$ is divisible by 3.

  Proof: $P(n)$ is true for $n = 0$ because $5^0 - 2^0 = 1 - 1 = 0$.

  If it is *not* true that $P(n)$ is true for every $n \geq 0$, then there are values of n greater than or equal to 0 for which $P(n)$ is false, and therefore there must be a smallest such value, say $n = k$.

  Since $P(0)$ is true, k must be at least 1.

  Therefore, $k - 1$ is at least 0, and since k is the smallest value for which P fails, $P(k - 1)$ is true.

  This means that $5^{k-1} - 2^{k-1}$ is a multiple of 3, say $3j$.

  Then, however,

  $$5^k - 2^k = 5 * 5^{k-1} - 2 * 2^{k-1} = 3 * 5^{k-1} + 2 * (5^{k-1} - 2^{k-1}) = 3 * 5^{k-1} + 2 * 3j$$

  This expression is divisible by 3, a contradiction.

  Therefore, $P(n)$ is true for every $n \geq 0$.

- Although an induction proof has the advantage that it does not involve proof by contradiction, both approaches are equally valid.

2.3 The Strong Principle of Mathematical Induction

- Some induction proofs require a better tool than the Principle of Mathematical Induction.

- **Example 2.11:** Integers Bigger Than 2 Have Prime Factorizations

  Let $P(n)$ be the statement that n is either prime or the product of two or more primes; we will try to prove that $P(n)$ is true for every $n \geq 2$.

  $P(2)$ is true, since 2 is a prime.
  The induction hypothesis is the statement that $k \geq 2$ and k is either prime or the product of two or more primes.
  The statement to be proved in the induction step is that $k + 1$ is either prime or the product of primes.
  If $k + 1$ is not prime, it has some positive integer divisor other than itself and 1. This means $k + 1 = r * s$ for some positive integers r and s, neither of which is 1 or $k + 1$.
  The next step would be to show that r and s are both either primes or products of primes. Unfortunately, the induction hypothesis refers only to k; it provides no information about r or s.

- Completing this proof requires the use of the Strong Principle of Mathematical Induction.

- **The Strong Principle of Mathematical Induction:** Suppose $P(n)$ is a statement involving an integer n. Then to prove that $P(n)$ is true for every $n \geq n_0$, it is sufficient to show these two things:

  1. $P(n_0)$ is true.
  2. For any $k \geq n_0$, if $P(n)$ is true for every n satisfying $n_0 \leq n \leq k$, then $P(k + 1)$ is true.

The Strong Principle of Mathematical Induction (Continued)

- The proof of Example 2.11 can now be finished. The steps are the same as before, but with a different induction hypthesis.

- **Example 2.11.** Proof by induction.

  To show: $P(n)$ is true for every $n \geq 2$, where $P(n)$ is the statement: n is either a prime or a product of two or more primes.

  Basis step. $P(2)$ is true because 2 is a prime.

  Induction hypothesis. $k \geq 2$, and for every n with $2 \leq n \leq k$, n is either prime or a product of two or more primes.

  Statement to be shown in induction step. $k + 1$ is either prime or a product of two or more primes.

  Proof of induction step. If $k + 1$ is prime, the statement $P(k + 1)$ is true. Otherwise, by definition of a prime, $k + 1 = r * s$, for some positive integers r and s, neither of which is 1 or $k + 1$. It follows that $2 \leq r \leq k$ and $2 \leq s \leq k$. Therefore, by the induction hypothesis, both r and s are either prime or the product of two or more primes. Therefore, their product $k + 1$ is the product of two or more primes, and $P(k + 1)$ is true.

- The strong principle of induction is also referred to as the principle of *complete* induction, or *course-of-values* induction.

- The strong principle of induction is equivalent to the ordinary induction principle.

- Neither induction principle can be proved using other standard properties of the natural numbers.

The Strong Principle of Mathematical Induction (Continued)

- The *well-ordering* principle for the natural numbers (every nonempty subset of $\mathcal{N}$ has a smallest element) can be proved using the strong principle of induction (and is actually equivalent to it).

- **Example 2.12:** The Well-ordering Principle for the Natural Numbers

  To prove: Every nonempty subset of $\mathcal{N}$ has a smallest element.

  Proof: In order to use induction, let $P(n)$ be the statement "Every subset of $\mathcal{N}$ containing n has a smallest element." The goal is now to prove that $P(n)$ is true for every $n \geq 0$.

  Basis step. $P(0)$ is the statement that every subset of $\mathcal{N}$ containing 0 has a smallest element, which is true because 0 is the smallest natural number.

  Induction hypothesis. $k \geq 0$, and for every n with $0 \leq n \leq k$, every subset of $\mathcal{N}$ containing n has a smallest element.

  Statement to be shown in induction step. Every subset of $\mathcal{N}$ containing $k + 1$ has a smallest element.

  Proof of induction step. Let A be any subset of $\mathcal{N}$ containing $k + 1$. If A contains no natural number less than $k + 1$, then $k + 1$ is the smallest element of A. Otherwise, A contains some natural number n with $n \leq k$. In this case, by the induction hypothesis, A contains a smallest element.

- It may not be obvious at the beginning of an induction proof whether the strong induction principle is required. When in doubt, use the strong version.

2.4.1 Recursive Definitions of Functions with Domain $\mathcal{N}$

- The factorial function is often defined in a recursive manner:

$$n! = \begin{cases} 1 & if\ n = 0 \\ n \times (n-1)! & if\ n > 0 \end{cases}$$

- Recursive definitions resemble proofs by mathematical induction, with a basis step and an induction step.

The Fibonacci Function

- **Example 2.13:** The Fibonacci Function

  The Fibonacci function f is usually defined as follows:

  $f(0) = 1$
  $f(1) = 1$
  for every $n \geq 1$, $f(n + 1) = f(n) + f(n - 1)$

  The definition of the Fibonacci function suggests the strong principle of induction, because the definition of $f(n + 1)$ involves not only $f(n)$ but $f(n - 1)$. This observation is useful in proving facts about f.

  To prove: For every $n \geq 0$, $f(n) \leq (5/3)^n$.

  Basis step. $f(0) \leq (5/3)^0$ is true, since $f(0)$ and $(5/3)^0$ are both 1.

  Induction hypothesis. $k \geq 0$, and for every n with $0 \leq n \leq k$, $f(n) \leq (5/3)^n$.

  Statement to show in induction step. $f(k + 1) \leq (5/3)^{k+1}$.

  Proof of induction step. If $k = 0$, then $f(k + 1) = f(1) = 1$, by definition, and in this case the inequality is clearly true. If $k \geq 0$, then we must use the recursive part of the definition, which is $f(k + 1) = f(k) + f(k - 1)$. Since both k and $k - 1$ are $\leq k$, we may apply the induction hypothesis to both terms, obtaining

  $$\begin{aligned} f(k + 1) &= f(k) + f(k - 1) \\ &\leq (5/3)^k + (5/3)^{k-1} \\ &= (5/3)^{k-1}(5/3 + 1) \\ &= (5/3)^{k-1}(8/3) \\ &= (5/3)^{k-1}(24/9) \\ &< (5/3)^{k-1}(25/9) \\ &= (5/3)^{k+1} \end{aligned}$$

The Union of *n* Sets

- **Example 2.14:** The Union of *n* Sets

  Suppose $A_1, A_2, \ldots$ are subsets of some universal set U. For each $n \geq 0$, the union of $A_1, A_2, \ldots, A_n$ can be defined as follows:

$$\bigcup_{i=1}^{0} A_i = \varnothing$$

  for every $n \geq 0$, $\bigcup_{i=1}^{n+1} A_i = \left(\bigcup_{i=1}^{n} A_i \right) \cup A_{n+1}$

  Similarly, the summation of $a_1, a_2, \ldots, a_n$ can be defined as follows:

$$\sum_{i=1}^{0} a_i = 0$$

  for every $n \geq 0$, $\sum_{i=1}^{n+1} a_i = \left(\sum_{i=1}^{n} a_i \right) + a_{n+1}$

- Recursion could be also used to define the intersection of *n* sets, the product of *n* numbers, the concatenation of *n* strings, the concatenation of *n* languages, and so forth.

Advantages of Recursive Definitions

- Consider the recursive definition of the concatenation of n languages, all of which are the same language L:

  $$L^0 = \{\Lambda\}$$
  for every $n \geq 0$, $L^{n+1} = L^n L$

  The non-recursive definition was

  $L^n = LL \cdots L$ (n factors in all)

- Advantages of the recursive definition:

  More consistent with the binary nature of concatenation
  More explicit about how the concatenation is done (more algorithmic)
  Provides a natural way of constructing proofs using mathematical induction

- There is a close relationship between recursive definitions and the principle of mathematical induction:

  Recursive definitions are useful in constructing induction proofs.
  Induction is the natural proof technique to use on objects defined recursively.

2.4.2 Recursive Definitions of Sets

- A recursive definition of a set specifies certain objects that are in the set to start with and describes one or more general methods for obtaining new elements of the set from existing ones.

- **Example 2.15:** Recursive Definition of L^*

  A recursive definition of a language L^*, where L is a language over Σ:

  1. $\Lambda \in L^*$.
  2. For any $x \in L^*$ and any $y \in L$, $xy \in L^*$.
  3. No string is in L^* unless it can be obtained by using rules 1 and 2.

  An alternative definition of L^*:

  1. $\Lambda \in L^*$.
  2. For any $x \in L$, $x \in L^*$.
  3. For any two elements x and y of L^*, $xy \in L^*$.
  4. No string is in L^* unless it can be obtained by using rules 1, 2, and 3.

- **Example 2.16:** Palindromes

  Let Σ be any alphabet. The language *pal* of *palindromes* over Σ can be defined as follows:

  1. $\Lambda \in pal$.
  2. For any $a \in \Sigma$, $a \in pal$.
  3. For any $x \in pal$ and any $a \in \Sigma$, $axa \in pal$.
  4. No string is in *pal* unless it can be obtained by using rules 1, 2, and 3.

Recursive Definitions of Sets (Continued)

- **Example 2.17:** Fully Parenthesized Algebraic Expressions

  Let $\Sigma = \{i, (,), +, -\}$. Below is a recursive definition of the language *AE* of fully parenthesized algebraic expressions involving the binary operators + and – and the identifier *i*.

  1. $i \in AE$.
  2. For any $x, y \in AE$, both $(x + y)$ and $(x - y)$ are elements of *AE*.
  3. No string is in *AE* unless it can be obtained by using rules 1 and 2.

  Some of the strings in *AE* are i, $(i + i)$, $(i - i)$, $((i + i) - i)$, and $((i - (i - i)) + i)$.

- **Example 2.18:** Finite Subsets of the Natural Numbers

  The set $\mathcal{F}$ of subsets of the natural numbers is defined as follows:

  1. $\varnothing \in \mathcal{F}$.
  2. For any $n \in \mathcal{N}$, $\{n\} \in \mathcal{F}$.
  3. For any A and B in $\mathcal{F}$, $A \cup B \in \mathcal{F}$.
  4. Nothing is in $\mathcal{F}$ unless it can be obtained by using rules 1, 2, and 3.

  It is easy to show that $\mathcal{F}$ is the collection of all finite subsets of $\mathcal{N}$.

- In a recursive definition, the last statement is present to make it explicit that no elements belong to the set other than the ones specified by the previous rules.

- It is also understood (but not usually stated) that no elements belong to the set unless they can be obtained by a *finite* number of applications of the rules.

- A less formal approach would be to have the last statement be "Nothing else is in $\mathcal{F}$" or to skip this statement altogether.

An Induction Proof Based on a Recursive Set Definition

- **Example 2.19:** An Induction Proof Involving a Language Defined Recursively

  Suppose that the language L is defined recursively as follows.

  1. $\Lambda \in L$.
  2. For any $y \in L$, both $0y$ and $0y1$ are in L.
  3. No string is in L unless it can be obtained from rules 1 and 2.

  L appears to be equal to the language A, where $A = \{0^i 1^j \mid i \geq j \geq 0\}$. To show that $L = A$, it is necessary to show that $A \subseteq L$ and $L \subseteq A$.

  To prove: $A \subseteq L$. In order to use induction, this statement will need to be rewritten as follows so that it involves an integer: For every $n \geq 0$, every $x \in A$ satisfying $|x| = n$ is an element of L.

  Basis step. Suppose that x is in A and $|x| = 0$. Then $x = \Lambda$, and statement 1 in the definition of L says that $\Lambda \in L$.

  Induction hypothesis. $k \geq 0$, and every x in A with $|x| \leq k$ is an element of L.

  Statement to show in induction step. Every x in A with $|x| = k + 1$ is an element of L.

  Proof of induction step. Suppose $x \in A$ and $|x| = k + 1$. Then $x = 0^i 1^j$, where $i \geq j \geq 0$, and $i + j = k + 1$.

  Case 1: If $i > j$, then $x = 0y$ for some $y \in A$. Since $|y| = k$, it follows from the induction hypothesis that $y \in L$; therefore, it follows from the first part of statement 2 in the definition of L that x is also an element of L.

  Case 2: If $i = j$, there must be at least one 0 and one 1 in x. Therefore, $x = 0y1$ for some y. Furthermore, $y \in A$, because $y = 0^{i-1}1^{j-1}$ and $i = j$. Since $|y| \leq k$, it follows from the induction hypothesis that $y \in L$. Since $x = 0y1$, we can use the second part of statement 2 in the definition of L to conclude that $x \in L$.

- Notice that the strong induction principle is needed because of Case 2, where y has length $k - 1$, not k.

2.5 Structural Induction

- There is a close correspondence between recursive definitions of functions on the set $\mathcal{N}$ and proofs by mathematical induction of properties of those functions.

- It is also possible to write induction proofs that are based on recursive set definitions, without having to introduce an integer for induction purposes.

- **Example 2.20:** Continuation of Example 2.19

  The language L is defined recursively as follows:

  1. $\Lambda \in L$.
  2. For every $x \in L$, both $0x$ and $0x1$ are in L.

  The language A is defined to be $\{0^i1^j \mid i \geq j \geq 0\}$. Example 2.19 used mathematical induction to prove $A \subseteq L$. The goal is now to prove $L \subseteq A$.

  To prove: For every $n \geq 0$, every $x \in L$ obtained by n applications of rule 2 is an element of A.

  Basis step. Suppose that $x \in L$ and x is obtained without using rule 2 at all. Then x must be Λ, and $\Lambda \in A$ because $\Lambda = 0^01^0$.

  Induction hypothesis. $k \geq 0$, and every string in L that can be obtained by k applications of rule 2 is an element of A.

  Statement to show in induction step. Any string in L that can be obtained by $k + 1$ applications of rule 2 is in A.

  Proof of induction step. Let x be an element of L that is obtained by $k + 1$ applications of rule 2. This means that either $x = 0y$ or $x = 0y1$, where in either case y is a string in L that can be obtained by using the rule k times. By the induction hypothesis, $y \in A$, so that $y = 0^i1^j$, with $i \geq j \geq 0$. Therefore, either $x = 0^{i+1}1^j$ or $x = 0^{i+1}1^{j+1}$, and in either case $x \in A$.

Structural Induction (Continued)

- The proof that $L \subseteq A$ can be simplified by removing the references to k and $k + 1$:

  The strings in L that can be obtained by $k + 1$ applications of rule 2 are of the form $0y$ and $0y1$, where y is can be obtained by k applications of rule 2. Therefore, the induction hypothesis can refer to any string y in L, and the statement to be proved can be that $0y$ and $0y1$ are in A.

  The basis step will refer to the string in rule 1.

- This version of mathematical induction is called *structural induction*. With structural induction, the underlying integer on which the induction is based is not mentioned explicitly. Instead, the steps of the proof follow the structure of the recursive definition directly.

- A proof of $L \subseteq A$ using structural induction:

  L is defined as follows:

  1. $\Lambda \in L$.
  2. For every $x \in L$, both $0x$ and $0x1$ are in L.

  A is defined to be $\{0^i1^j \mid i \geq j \geq 0\}$.

  To prove: $L \subseteq A$.

  Basis step. $\Lambda \in A$ is true, because $\Lambda = 0^0 1^0$.

  Induction hypothesis. The string $y \in L$ is an element of A.

  Statement to show in induction step. Both $0y$ and $0y1$ are elements of A.

  Proof of induction step. Since $y \in A$, $y = 0^i1^j$, with $i \geq j \geq 0$. Therefore, $0y = 0^{i+1}1^j$, and $0y1 = 0^{i+1}1^{j+1}$, and both strings are in A.

- When a set L is defined recursively, structural induction can be used to show that every element of L has some property. In the previous example, the "property" is that of belonging to the set A.

Structural Induction (Continued)

- **Example 2.21:** Another Property of Fully Parenthesized Algebraic Expressions

  The language *AE* was defined in Example 2.17 as follows:

  1. $i \in AE$.
  2. For any x and y in *AE*, both $(x + y)$ and $(x – y)$ are in *AE*.
  3. No other strings are in *AE*.

  To prove: No string in *AE* contains the substring)(.

  Basis step. The string i does not contain the substring)(.

  Induction hypothesis. x and y do not contain the substring)(.

  Statement to show in induction step. Neither $(x + y)$ nor $(x – y)$ contains the substring)(.

  Proof of induction step. In both the expressions $(x + y)$ and $(x – y)$, the symbol preceding x is not), the symbol following x is not (, the symbol preceding y is not), and the symbol following y is not (. Therefore, the only way)(could appear would be for it to occur in x or y separately.

- For simplicity, the induction hypothesis is weaker than it could be, because it does not require x and y to be in *AE*.

Structural Induction (Continued)

- **Example 2.22:** The Language of Strings with More *a*'s than *b*'s

  Suppose that the language $L \subseteq \{a, b\}^*$ is defined as follows:

  1. $a \in L$.
  2. For any $x \in L$, $ax \in L$.
  3. For any x and y in L, all the strings bxy, xby, and xyb are in L.
  4. No other strings are in L.

  To prove: Every element of L has more *a*'s than *b*'s.

  Proof: By structural induction. Again, the induction step can be simplified by proving something even stronger than necessary.

  Basis step. The string *a* has more *a*'s than *b*'s.

  Induction hypothesis. *x* and *y* are strings containing more *a*'s than *b*'s.

  Statement to show in induction step. Each of the strings *ax*, *bxy*, *xby*, and *xyb* has more *a*'s than *b*'s.

  Proof of induction step. The string *ax* clearly has more *a*'s than *b*'s, since *x* does. Since both *x* and *y* have more *a*'s than *b*'s, *xy* has at least two more *a*'s than *b*'s, and therefore any string formed by inserting one more *b* still has at least one more *a* than *b*'s.

Recursive Definitions of Functions on Sets

- A recursive definition of a set can also provide a useful way of defining a function on the set.

- **Example 2.24:** Recursive Definitions of the Length and Reverse Functions

  Recursive definition of the length function on Σ^*:

  1. $|\Lambda| = 0$.
  2. For any $x \in \Sigma^*$ and any $a \in \Sigma$, $|xa| = |x| + 1$.

  Recursive definition of the *rev* function on Σ^* ($rev(x)$ is often written x^r):

  1. $\Lambda^r = \Lambda$.
  2. For any $x \in \Sigma^*$ and any $a \in \Sigma$, $(xa)^r = ax^r$.

  To prove: For every x and y in Σ^*, $|xy| = |x| + |y|$

  Proof: The approach will be to use structural induction to prove that the following statement is true for every string y: for every x, $|xy| = |x| + |y|$.

  Basis step: The statement is true when $y = \Lambda$, because for every x,

  $|x\Lambda| = |x| = |x| + 0 = |x| + |\Lambda|$

  Induction hypothesis: For every x, $|xy| = |x| + |y|$.

  Statement to show in induction step. For every x, $|x(ya)| = |x| + |ya|$, where a is an arbitrary element of Σ.

  Proof of induction step.

$\|x(ya)\|$	$= \|(xy)a\|$	(because concatenation is associative)
	$= \|xy\| + 1$	(by the recursive definition of the length function)
	$= (\|x\| + \|y\|) + 1$	(by the induction hypothesis)
	$= \|x\| + (\|y\| + 1)$	(because addition is associative)
	$= \|x\| + \|ya\|$	(by the definition of the length function)