

Section 3.2 – Recursive Definitions and Structural Induction

Recursive Algorithms

Definition

A *recursive* or *inductive definition* of a function consists of two steps.

1. **BASIS STEP:** Specify the value of the function at zero.
2. **RECURSIVE STEP:** Give a rule for finding its value at an integer from its values at smaller integers.

Example

Suppose f is defined by:

$$f(0) = 3$$

$$f(n+1) = 2f(n) + 3$$

Find $f(1), f(2), f(3), f(4)$

Solution

$$f(1) = 2f(0) + 3 = 2 \cdot 3 + 3 = 9$$

$$f(2) = 2f(1) + 3 = 2 \cdot 9 + 3 = 21$$

$$f(3) = 2f(2) + 3 = 2 \cdot 21 + 3 = 45$$

$$f(4) = 2f(3) + 3 = 2 \cdot 45 + 3 = 93$$

Example

Give a recursive definition of the factorial function $n!$

Solution

$$f(0) = 1$$

$$f(n+1) = (n+1) \cdot f(n)$$

These 2 equations define $n!$

Example

Give a recursive definition of a^n , where a is a nonzero real number and n is a nonnegative integer.

Solution

$$a^0 = 1$$

$$a^{n+1} = a \cdot a^n \text{ for } n = 0, 1, 2, 3, \dots$$

These 2 equations define a^n for all nonnegative integers n .

Example

Give a recursive definition of: $\sum_{k=0}^n a_k$

Solution

The first part of the definition is $\sum_{k=0}^0 a_k = a_0$

The second part is $\sum_{k=0}^{n+1} a_k = \left(\sum_{k=0}^n a_k \right) + a_{n+1}$

Fibonacci Numbers

Example

The Fibonacci numbers are defined as follows:

$$f_0 = 0$$

$$f_1 = 1$$

$$f_n = f_{n-1} + f_{n-2}$$

Find f_2, f_3, f_4, f_5

Solution

$$f_2 = f_1 + f_0 = 1 + 0 = \underline{1}$$

$$f_3 = f_2 + f_1 = 1 + 1 = \underline{2}$$

$$f_4 = f_3 + f_2 = 2 + 1 = \underline{3}$$

$$f_5 = f_4 + f_3 = 3 + 2 = \underline{5}$$

Example

Show that whenever $n \geq 3$, $f_n = \alpha^{n-2}$, where $\alpha = \frac{1+\sqrt{5}}{2}$

Solution

Basis Step: $n = 3$, $\alpha < 2 = f_3$

$$n = 4, \alpha^2 = \left(\frac{1+\sqrt{5}}{2} \right)^2 = \frac{6+2\sqrt{5}}{4} = \frac{3+\sqrt{5}}{2} < 3 = f_4$$

Inductive Step: Assume that P_k holds $f_k = \alpha^{k-2}$

We need to prove that P_{k+1} is true, that is $f_{k+1} = \alpha^{(k+1)-2} = \alpha^{k-1}$ is also true.

$$\begin{aligned}\alpha^{k-1} &= \alpha^2 \cdot \alpha^{k-3} & \alpha^2 &= \left(\frac{1+\sqrt{5}}{2} \right)^2 = \frac{3+\sqrt{5}}{2} = \frac{2+1+\sqrt{5}}{2} = 1 + \frac{1+\sqrt{5}}{2} = 1 + \alpha \\ &= (\alpha + 1) \cdot \alpha^{k-3} \\ &= \alpha^{k-2} + \alpha^{k-3}\end{aligned}$$

By the inductive hypothesis, we have $f_k > \alpha^{k-2}$ $f_{k-1} > \alpha^{k-3}$

Therefore. $f_{k+1} = f_k + f_{k-1} > \alpha^{k-2} + \alpha^{k-3} = \alpha^{k-1}$

Hence, P_{k+1} is true. This completes the proof.

Lamé's Theorem

Let a and b be positive integers with $a \geq b$. Then the number of divisions used by the Euclidian algorithm to find $\gcd(a, b)$ is less than or equal to five times the number of decimal digits in b .

Proof

When we use the Euclidian algorithm to find $\gcd(a, b)$ with $a \geq b$,

n divisions are used to obtain (with $a = r_0, b = r_1$):

$$\begin{aligned} r_0 &= r_1 q_1 + r_2 & 0 \leq r_2 \leq r_1 \\ r_1 &= r_2 q_2 + r_3 & 0 \leq r_3 \leq r_2 \\ &\vdots \\ r_{n-2} &= r_{n-1} q_{n-1} + r_n & 0 \leq r_n \leq r_{n-1} \\ r_{n-1} &= r_n q_n \end{aligned}$$

Since each quotient q_1, q_2, \dots, q_{n-1} is at least 1 and $q_n \geq 2$:

$$\begin{aligned} r_n &\geq 1 = f_2 \\ r_{n-1} &\geq 2r_n \geq 2f_2 = f_3 \\ r_{n-2} &\geq r_{n-1} + r_n \geq f_3 + f_2 = f_4 \\ &\vdots \\ r_2 &\geq r_3 + r_4 \geq f_{n-1} + f_{n-2} = f_n \\ b = r_1 &\geq r_2 + r_3 \geq f_n + f_{n-1} = f_{n+1} \end{aligned}$$

It follows that if n divisions are used by the Euclidian algorithm to find $\gcd(a, b)$ with $a \geq b$, then

$$b \geq f_{n+1}. \quad f_{n+1} \geq \alpha^{n-1}, \text{ for } n > 2, \text{ where } \alpha = \frac{1+\sqrt{5}}{2}. \text{ Therefore, } b > \alpha^{n-1}.$$

Because $\log \alpha \approx 0.208 > \frac{1}{5}$, $\log b > (n-1) \log \alpha > \frac{n-1}{5}$. Hence, $n-1 < 5 \cdot \log b$

Suppose that b has k decimal digits. Then $b < 10^k$ and $\log b < k$. It follows that $n-1 < 5k$ and since k is an integer, $n \leq 5k$.

As a consequence of Lamé's Theorem, $O(\log b)$ divisions are used by the Euclidian algorithm to find $\gcd(a, b)$ whenever $a > b$.

Recursively Defined Sets and Structures

Recursive definitions of sets have two parts:

- The **basis step** specifies an initial collection of elements.
- The **recursive step** gives the rules for forming new elements in the set from those already known to be in the set.

Sometimes the recursive definition has an *exclusion rule*, which specifies that the set contains nothing other than those elements specified in the basis step and generated by applications of the rules in the recursive step.

We will always assume that the exclusion rule holds, even if it is not explicitly mentioned.

We will later develop a form of induction, called *structural induction*, to prove results about recursively defined sets.

Example

Consider the subset S of the set of integers recursively defined by

Basis step: $3 \in S$.

Recursive step: If $x \in S$ and $y \in S$, then $x + y$ is in S .

Initially 3 is in S , then $3 + 3 = 6$, then $3 + 6 = 9$, etc.

Example

Consider the subset N of the set of natural numbers recursively defined by

Basis step: $0 \in N$.

Recursive step: If n is in N , then $n + 1$ is in N .

Initially 0 is in S , then $0 + 1 = 1$, then $1 + 1 = 2$, etc.

Strings

Definition

The set Σ^* of *strings* over the alphabet Σ :

Basis step: $\lambda \in \Sigma^*$ (λ is the empty string)

Recursive step: If $w \in \Sigma^*$ and $x \in \Sigma$, then $wx \in \Sigma^*$.

Example

If $\Sigma = \{0,1\}$, the strings in Σ^* are the set of all bit strings, $\lambda, 0, 1, 00, 01, 10, 11$, etc.

Example

If $\Sigma = \{a,b\}$, show that aab is in Σ^* .

Since $\lambda \in \Sigma^*$ and $a \in \Sigma$, $a \in \Sigma^*$.

Since $a \in \Sigma^*$ and $a \in \Sigma$, $aa \in \Sigma^*$.

Since $aa \in \Sigma^*$ and $b \in \Sigma$, $aab \in \Sigma^*$.

String Concatenation

Definition

Two strings can be combined via the operation of *concatenation*. Let Σ be a set of symbols and Σ^* be the set of strings formed from the symbols in Σ . We can define the concatenation of two strings, denoted by \cdot , recursively as follows.

Basis step: If $w \in \Sigma^*$, then $w \cdot \lambda = w$.

Recursive step: If $w_1 \in \Sigma^*$ and $w_2 \in \Sigma^*$ and $x \in \Sigma$, then $w_1 \cdot (w_2 x) = (w_1 \cdot w_2) x$.

✓ Often $w_1 \cdot w_2$ is written as $w_1 w_2$.

✓ If $w_1 = abra$ and $w_2 = cadabra$, the concatenation $w_1 w_2 = abracadabra$.

Length of a String

Example

Give a recursive definition of $l(w)$, the length of the string w .

Solution

The length of a string can be recursively defined by:

$l(w) = 0$;

$l(wx) = l(w) + 1$ if $w \in \Sigma^*$ and $x \in \Sigma$.

Well-Formed Formulae in Propositional Logic

Definition

The set of *well-formed formulae* in propositional logic involving **T**, **F**, propositional variables, and operators from the set $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$.

Basis step: **T**, **F**, and s , where s is a propositional variable, are well-formed formulae.

Recursive step: If E and F are well formed formulae, then $(\neg E)$, $(E \wedge F)$, $(E \vee F)$, $(E \rightarrow F)$, $(E \leftrightarrow F)$, are well-formed formulae.

Examples

$((p \vee q) \rightarrow (q \wedge \mathbf{F}))$ is a well-formed formula.

$pq \wedge$ is not a well formed formula.

Rooted Trees

Definition

The set of *rooted trees*, where a rooted tree consists of a set of vertices containing a distinguished vertex called the *root*, and edges connecting these vertices, can be defined recursively by these steps:

Basis step: A single vertex r is a rooted tree.

Recursive step: Suppose that T_1, T_2, \dots, T_n are disjoint rooted trees with roots r_1, r_2, \dots, r_n ,

respectively. Then the graph formed by starting with a root r , which is not in any of the rooted trees T_1, T_2, \dots, T_n , and adding an edge from r to each of the vertices

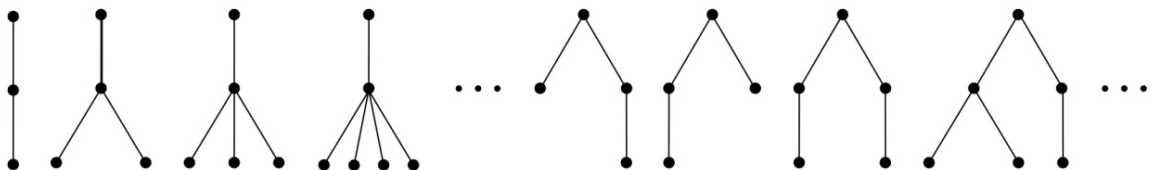
r_1, r_2, \dots, r_n , is also a rooted tree.

Basis step •

Step 1



Step 2



Definition

The set of **extended binary trees** can be defined recursively by these steps:

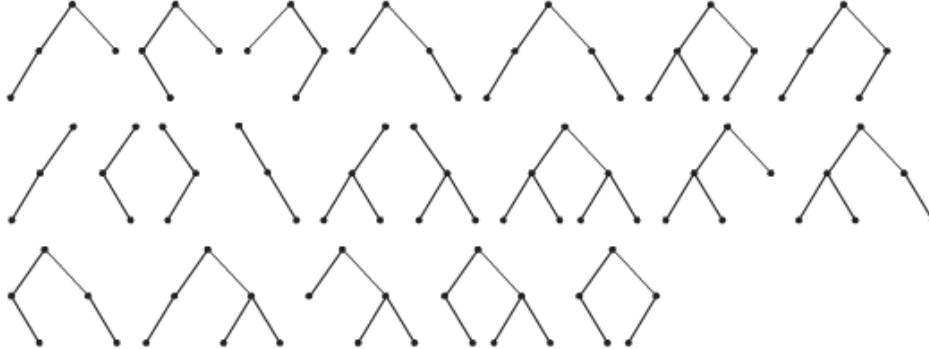
Basis step: The empty set is an extended binary tree.

Recursive step: If T_1 and T_2 are disjoint extended binary trees, there is an extended binary tree, denoted by $T_1 \cdot T_2$, consisting of a root r together with edges connecting the root to each of the roots of the left subtree T_1 and the right subtree T_2 when these trees are nonempty.

Basis step \emptyset

Step 1 

Step 2 

Step 3 

Extended Binary Trees

Definition

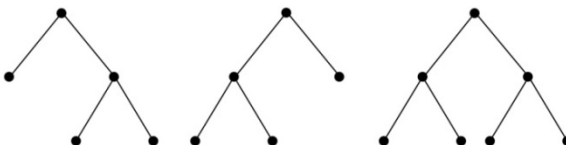
The set of **full binary trees** can be defined recursively by these steps:

Basis step: There is a full binary tree consisting only of a single vertex r .

Recursive step: If T_1 and T_2 are disjoint full binary trees, there is a full binary tree, denoted by $T_1 \cdot T_2$, consisting of a root r together with edges connecting the root to each of the roots of the left subtree T_1 and the right subtree T_2 .

Basis step 

Step 1 

Step 2 

Full Binary Trees

Structural Induction

Example

Show that the set S defined by specifying that $3 \in S$ and that if $x \in S$ and $y \in S$, then $x + y$ is in S , is the set of all positive integers that are multiples of 3.

Solution

Let A be the set of all positive integers divisible by 3.

To prove that $A = S$, show that A is a subset of S and S is a subset of A .

$A \subseteq S$: Let $P(n)$ be the statement that $3n$ belongs to S .

Basis step: $3 \cdot 1 = 3 \in S$, by the first part of recursive definition.

Inductive step: Assume $P(k)$ is true. By the second part of the recursive definition, if $3k \in S$, then since $3 \in S$, $3k + 3 = 3(k + 1) \in S$. Hence, $P(k + 1)$ is true.

$S \subseteq A$:

Basis step: $3 \in S$ by the first part of recursive definition, and $3 = 3 \cdot 1$.

Inductive step: The second part of the recursive definition adds $x + y$ to S , if both x and y are in S . If x and y are both in A , then both x and y are divisible by 3. It follows that $x + y$ is divisible by 3.

Full Binary Trees

Definition

The *height* $h(T)$ of a full binary tree T is defined recursively as follows:

Basis step: The height of a full binary tree T consisting of only a root r is $h(T) = 0$.

Recursive step: If T_1 and T_2 are full binary trees, then the full binary tree $T = T_1 T_2$ has height

$$h(T) = 1 + \max(h(T_1), h(T_2)).$$

The number of vertices $n(T)$ of a full binary tree T satisfies the following recursive formula:

Basis step: The number of vertices of a full binary tree T consisting of only a root r is $n(T) = 1$.

Recursive step: If T_1 and T_2 are full binary trees, then the full binary tree $T = T_1 T_2$ has the number

$$\text{of vertices } n(T) = 1 + n(T_1) + n(T_2).$$

Theorem

If T is a full binary tree, then $n(T) \leq 2^{h(T)+1} - 1$

Proof

Use structural induction.

Basis step: The result holds for a full binary tree consisting only of a root, $n(T) = 1$ and $h(T) = 0$.

Hence, $n(T) = 1 \leq 2^{0+1} - 1 = 1$.

Recursive step: Assume $n(T_1) \leq 2^{h(T_1)+1} - 1$ and also $n(T_2) \leq 2^{h(T_2)+1} - 1$ whenever T_1 and T_2 are full binary trees.

$$n(T) = 1 + n(T_1) + n(T_2) \quad (\text{by recursive formula of } n(T))$$

$$\leq 1 + \left(2^{h(T_1)+1} - 1\right) + \left(2^{h(T_2)+1} - 1\right) \quad (\text{by inductive hypothesis})$$

$$\leq 2 \cdot \max\left(2^{h(T_1)+1}, 2^{h(T_2)+1}\right) - 1$$

$$= 2 \cdot 2^{\max(h(T_1)+1, h(T_2))+1} - 1 \quad \left(\max\left(2^x, 2^y\right) = 2^{\max(x,y)}\right)$$

$$= 2 \cdot 2^{h(T)+1} - 1 \quad (\text{by recursive definition of } h(T))$$

$$= 2^{h(T)+1+1} - 1$$

Exercises **Section 3.2 – Recursive Definitions and Structural Induction**

1. Find $f(1), f(2), f(3)$, and $f(4)$ if $f(n)$ is defined recursively by $f(0) = 1$ and for $n = 0, 1, 2, \dots$
 - a) $f(n+1) = f(n) + 2$
 - b) $f(n+1) = 3f(n)$
 - c) $f(n+1) = 2^{f(n)}$
 - d) $f(n+1) = f(n)^2 + f(n) + 1$
2. Find $f(1), f(2), f(3), f(4)$ and $f(5)$ if $f(n)$ is defined recursively by $f(0) = 3$ and for $n = 0, 1, 2, \dots$
 - a) $f(n+1) = -2f(n)$
 - b) $f(n+1) = 3f(n) + 7$
 - c) $f(n+1) = 3^{f(n)/3}$
 - d) $f(n+1) = f(n)^2 - 2f(n) - 2$
3. Find $f(2), f(3), f(4)$ and $f(5)$ if $f(n)$ is defined recursively by $f(0) = f(1) = 1$ and for $n = 1, 2, \dots$
 - a) $f(n+1) = f(n) - f(n-1)$
 - b) $f(n+1) = f(n)f(n-1)$
 - c) $f(n+1) = f(n)^2 + f(n-1)^3$
 - d) $f(n+1) = f(n) / f(n-1)$
4. Determine whether each of these proposed definitions is a valid recursive definition of a function f from the set of nonnegative integers to the set of integers. If f is well defined, find a formula for $f(n)$ when n is nonnegative integer and prove that your formula is valid.
 - a) $f(0) = 0, f(n) = 2f(n-2)$ for $n \geq 1$
 - b) $f(0) = 1, f(n) = -f(n-1)$ for $n \geq 1$
 - c) $f(0) = 1, f(n) = f(n-1) - 1$ for $n \geq 1$
 - d) $f(0) = 2, f(1) = 3, f(n) = f(n-1) - 1$ for $n \geq 2$
 - e) $f(0) = 1, f(1) = 2, f(n) = 2f(n-2)$ for $n \geq 2$
 - f) $f(0) = 1, f(1) = 0, f(2) = 2, f(n) = 2f(n-3)$ for $n \geq 3$
 - g) $f(0) = 0, f(1) = 1, f(n) = 2f(n+1)$ for $n \geq 2$
 - h) $f(0) = 0, f(1) = 1, f(n) = 2f(n-1)$ for $n \geq 2$
 - i) $f(0) = 2, f(n) = f(n-1)$ if n is odd and $n \geq 1$ and $f(n) = 2f(n-2)$ if n is even and $n \geq 2$
 - j) $f(0) = 1, f(n) = 3f(n-1)$ if n is odd and $n \geq 1$ and $f(n) = 9f(n-2)$ if n is even and $n \geq 2$

5. Give a recursive definition of the sequence $\{a_n\}$, $n = 1, 2, 3, \dots$ if
- a) $a_n = 6n$ b) $a_n = 2n + 1$ c) $a_n = 10^n$ d) $a_n = 5$
e) $a_n = 4n - 2$ f) $a_n = 1 + (-1)^n$ g) $a_n = n(n + 1)$ h) $a_n = n^2$
6. Prove that $f_1^2 + f_2^2 + \dots + f_n^2 = f_n f_{n+1}$ when n is a positive integer and f_n is the n th Fibonacci number.
7. Prove that $f_1 + f_3 + \dots + f_{2n-1} = f_{2n}$ when n is a positive integer and f_n is the n th Fibonacci number.
8. Give a recursive definition of
- a) The set of odd positive integers
b) The set of positive integers powers of 3
c) The set of polynomial with integer coefficients
d) The set of even integers
e) The set of positive integers congruent to 2 modulo 3.
f) The set of positive integers not divisible by 5
9. Let S be the subset of the set of ordered pairs of integers defined recursively by
Basis step: $(0, 0) \in S$.
Recursive step: If $(a, b) \in S$, then $(a + 2, b + 3) \in S$ and $(a + 3, b + 2) \in S$
- a) List the elements of S produced by the first five applications of the recursive definition.
b) Use strong induction on the number of applications of the recursive step of the definition to show that $5 \mid a + b$ when $(a, b) \in S$.
c) Use structural induction to show that $5 \mid a + b$ when $(a, b) \in S$.
10. Let S be the subset of the set of ordered pairs of integers defined recursively by
Basis step: $(0, 0) \in S$.
Recursive step: If $(a, b) \in S$, then $(a, b + 1) \in S$, $(a + 1, b + 1) \in S$ and $(a + 2, b + 1) \in S$
- a) List the elements of S produced by the first five applications of the recursive definition.
b) Use strong induction on the number of applications of the recursive step of the definition to show that $a \leq 2b$ whenever $(a, b) \in S$.
c) Use structural induction to show that $a \leq 2b$ whenever $(a, b) \in S$.