Immagine che contiene testo, Carattere, schermata, documento

Descrizione generata automaticamenteTheoretical definitions

Immagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamenteImmagine che contiene testo, ricevuta, Carattere, bianco

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, schermata, documento

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, lettera, carta

Descrizione generata automaticamenteRecursiveness

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamenteImmagine che contiene testo, ricevuta, Carattere, algebra

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, documento

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, documento

Descrizione generata automaticamenteNon-computable functions

Immagine che contiene testo, Carattere, bianco, schermata

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, schermata, bianco

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, algebra

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Carattere, diagramma

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, algebra

Descrizione generata automaticamenteSmn-theorem

Immagine che contiene testo, Carattere, schermata, algebra

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, schermata, documento

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamenteSecond recursion theorem

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, ricevuta, bianco

Descrizione generata automaticamentePrimitive recursion

To prove that the given function f is primitive recursive, we will use the fact that the functions x^2, x + 3, and the characteristic function of "x is a multiple of 3" are all primitive recursive.

Let χ\_3(x) be the characteristic function of the predicate "x is a multiple of 3", defined as:

χ\_3(x) = sg(rm(3,x))

where rm(x,y) is the remainder function and sg(x) is the sign function, both known to be primitive recursive.

Now, we can express f as:

f(x) = χ\_3(x) · x^2 + (1 - χ\_3(x)) · (χ\_1(rm(3,x)) · (x + 3) + (1 - χ\_1(rm(3,x))) · x!)

where χ\_1 is the characteristic function of the predicate "x has a rest of 1 when dividing by 3".

Since the functions x^2, x + 3, x!, rm, sg, and the characteristic functions χ\_3 and χ\_1 are all primitive recursive, and primitive recursive functions are closed under composition and operations like addition and multiplication, we conclude that f is also primitive recursive.

Additional Examples:

1. The Collatz function C(x), defined as:

C(x) = x/2 if x is even

C(x) = 3x + 1 if x is odd

This function can be expressed using primitive recursion as:

C(0) = 0

C(x+1) = (1 - sg(rm(2,x+1))) · ((x+1)/2) + sg(rm(2,x+1)) · (3(x+1) + 1)

2. The log\_2 function, which computes the floor of the base-2 logarithm of its input:

log\_2(x) = 0 if x = 1

log\_2(x) = 1 + log\_2(x/2) if x > 1

This function can be defined by primitive recursion as follows:

log\_2(0) = 0

log\_2(x+1) = (1 - sg(x)) · 0 + sg(x) · (1 + log\_2((x+1)/2))

Other additional examples:

Example 1: The Fibonacci function

The Fibonacci function F(n) is defined as follows:

F(0) = 0

F(1) = 1

F(n+2) = F(n) + F(n+1)

To show F(n) is primitive recursive, we utilize the fact that primitive recursion allows defining a function in terms of multiple previous values. Define the helper function G:

G(x,0) = (0,1)

G(x,y+1) = (π₂(G(x,y)), π₁(G(x,y))+π₂(G(x,y)))

where π₁((a,b)) = a and π₂((a,b)) = b.

Now F can be defined using G as:

F(0) = 0

F(y+1) = π₁(G(0,y))

Since G is defined by primitive recursion from initial functions, projection functions π₁ and π₂, and arithmetic operations, it is primitive recursive. Therefore, F is also primitive recursive by composition.

Example 2: The exponential function a^n

For fixed a, the function f(n) = a^n can be defined by primitive recursion:

a⁰ = 1

a^(n+1) = a^n · a = f(n) · a

Using this, we can define the general case f(a,n) = a^n as:

f(0,n) = 0^n = 0 for n>0, 1 for n=0

f(a+1,0) = (a+1)⁰ = 1

f(a+1,n+1) = f(a+1,n) · (a+1)

So f(a,n) = a^n is primitive recursive.

Example 3: The binomial coefficient C(n,k)

The binomial coefficient C(n,k) counts the ways to choose k items from n items. It satisfies:

C(n,0) = C(n,n) = 1

C(n+1,k+1) = C(n,k) + C(n,k+1)

We can define it by nested primitive recursion:

h(0,k) = 1 if k=0, 0 otherwise

h(n+1,0) = 1

h(n+1,k+1) = h(n,k) + h(n,k+1)

Then C(n,k) = h(n,k) is primitive recursive.

In summary, the Fibonacci numbers, exponential function a^n, and binomial coefficients are all primitive recursive. This was shown by carefully defining them using the initial functions, projection functions, and the operations of primitive recursion and composition. The key is to break them down into simpler functions that are already known to be primitive recursive.