In this context, we need to know the total recursive, partial recursive, and primitive recursive function.

11.3.1.1 Partial and Total Function

A partial function f from a set A to set B is defined as a function from A' to B where A' is a subset of A. For all $x \in A$, there may exist $f(x) = y \in B$ or f(x) is undefined.

Example: Let A and B be two sets of positive integer numbers. A function f(x) is defined as x . The relation $A \to B$ only exists if $x \in A$ is a perfect square such as 4, 9, 16, 25, . . . etc. f(16) is defined but f(20) is undefined.

If $A' = A$, then the function f(x) is called a total function.

Example: $f(x) = x + 1$, where $x \in$ the set of integer numbers is a total function as f(x) is defined for all values of x.

11.3.1.2 Partial Recursive Function

A function computed by the Turing machine that need not halt for all input is called a partial recursive function. A partial recursive function is allowed to have an infinite loop for some input values.

11.3.1.3 Total Recursive Function

Partial recursive functions for which the Turing machine always halts are called the total recursive function. The total recursive function always returns a value for all possible input values.

From the discussion, it is clear that the total recursive function is a subset of the partial recursive function.

Example: Prove that the addition of two positive integers is a total recursive function.

Solution: $f(x, y) = x + y$, where $x, y \in$ set of positive integer numbers.
$f(x, 0) = x + 0 = x$ is the base condition.

$$f(x, y + 1) = x + y + 1 = f(x, y) + 1$$

Here, recursion occurs.

Thus, the function of the addition of two positive integers is recursive. The function is defined (returns a value) for all value of x, y, which proves it a total recursive.

11.3.1.4 Primitive Recursive Function

Primitive recursive functions are a subset of the total recursive function which can be obtained by a fi nite number of operations of composition and recursion from the initial functions (zero and successor function).

Primitive recursion is defined for $f(x_1, x_2, ..., x_n)$ as $f() = g(x_1, x_2, ..., x_n - 1)$ if $x_n = 0$

$$= h(x_1, x_2, ..., x_n, f(x_1, x_6, ..., x_n - 1)) \text{ if } x_n > 0$$

where g and h are primitive recursive functions.

Example: Every primitive recursive function is Turing computable.

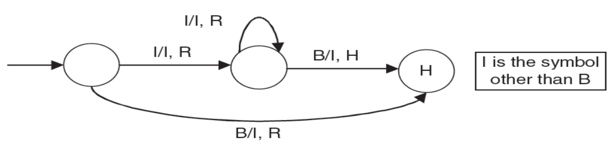
Solution: The primitive recursive function consists of the initial function (zero function, successor function, and projection function), composition, and recursion. If there exists a Turing machine for all these, then there exists a Turing machine for the primitive recursive function.

1. Turing machine for zero function

$Z(x) = 0$, for all $x \in N$, where N is the set of natural numbers

It is nothing but an eraser. There exists a Turing machine for an eraser.

1 picture

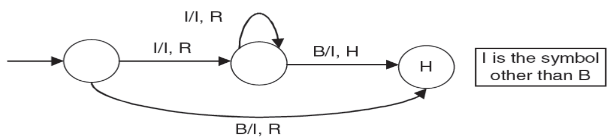


2. Turing machine for successor function

$S(x) = x + 1, forall x \in N$ where N is the set of natural numbers

The function adds a '1' with the value of x. If one blank symbol at the right hand side is replaced by '1', then the machine can be easily designed.

2 picture



3. A projection function is denoted as $P_i^n(x_1, x_2, x_3, ...x_n) = x_i$.

A Turing machine can be designed which takes input $Bx_1Bx_2Bx_3B......Bx_nB$ and produces the output Bx_iB.

4. A Turing Machine can be designed for *composition* of different functions by combining the respective Turing machines for each of the function in the order in which the functions appear.

5. Turing machine for $recursive functions$ can be designed by combining the Turing Machine designed for the simple function with multiple call and the Turing Machine designed for the Base function as termination point.

6. A final TM can be designed using the TMs for steps (4) and (5).

Hence, it is proved that every primitive recursive function is Turing computable.

Example 11.1 `Prove that the function` $f(x, y) = x+y$ `where` x, y `are positive integers is primitive recursive.`

Solution:

$$f(x, y) = x + y$$
$$\Rightarrow f(x, 0) = x = Z(x)$$
$$f(x, y + 1) = x + y + 1 = f(x, y) + 1 = S(x, f(x, y))$$

The function can be obtained by composition and recursion from the zero and successor functions in a finite number of steps. Thus, it is primitive recursive.
Let f(4, 3) be given.

$$f(4,3) = S(f(4,2)$$

$= f(4,2) + 1$
$= S(f(4,1)) + 1$
$= f(4,1) + 1 + 1$
$= S(f(4,0) + 1 + 1$
$= f(4,0) + 1 + 1 + 1$
$= 4 + 1 + 1 + 1$
$= 7$

Example 11.2 Prove that the function $f(x,y) = x-y$, where x, y are positive integers, is primitive recursive.

Solution:
$$f(x,y) = x - y$$
$$\Rightarrow f(x,0) = x = Z(x)$$
$$f(x,y+1) = x-(y+1) = x-y-1 = f(x,y)-1 = S(x, f(x,y))$$

The function can be obtained by composition and recursion from the zero and successor functions in a finite number of steps. Thus, it is primitive recursive.
Let f(4, 2) be given.

$$f(4,2) = S(f(4,1))$$

$= f(4,1) - 1$
$= S(f(4,0)) - 1$
$= f(4,0) - 1 - 1$
$= 4 - 1 - 1$
$= 2$

Example 11.3 Prove that the function $f(x, y) = x*y$, where x, y are positive integers, is primitive recursive.

Solution:
$$f(x, y) = x * y$$
$$\Rightarrow f(x, 0) = 0 = Z(x)$$
$$f(x, y+1) = x*(y+1) = x*y+x = f(x, y)+x = Add(x, f(x, y))$$

We have already proved that addition is primitive recursive, thus multiplication is primitive recursive.
Let f(3, 2) be given.

$$f(3, 2) = S(f(3, 1))$$
$= f(3, 1) + 3[AsS(f(x, y)) = f(x, y) + x]$
$= S(f(3, 0)) + 3$
$= f(3, 0) + 3 + 3$
$= 0 + 3 + 3$
$= 6$

Example 11.4 Prove that the function $f(x, y) = x^y$, where x, y are positive integers, is primitive recursive.

Solution:

$$f(x, y) = x^y$$
$$\Rightarrow f(x, 0) = x^0 = 1 = Z(x)$$
$$f(x, y+1) = x^y + 1 = x^y * x = f(x, y) * x = Mult(x, f(x, y))$$

We have already proved that multiplication is primitive recursive, thus exponential is primitive recursive.
Let f(4, 2) be given.

$$f(4, 2) = S(f(4, 1))$$

$= f(4, 1) * 4$
$= S(f(4, 0)) * 4$
$= f(4, 0) * 4 * 4$
$= 1 * 4 * 4$
$= 16$

Example 11.5 Prove that the function $f(x) = x/2$ if x is even

$= (x - 2)/2$ if $x is$ odd

where x is a positive integer which is primitive recursive.

Solution:
$f(x) = x/2,$ if x is even
$f(0) = 0/2 = 0$

If x is even, then x + 1 is odd.

$$f(x+1) = (x+1-2)/2$$

$= x/2 - 1/2$

$= f(x) - 1/2 = S(x, f(x))$

If x is odd, then $x+1$ is even.

$$f(x+1) = (x+1)/2$$

$= (x-2)/2 + 3/2$

$= f(x) + 3/2$

$= S(f(x))$

The given function can be rewritten as

$$S(f(x)) = f(x) - 1/2, \quad \text{if x is even}$$

$= f(x) + 3/2, \quad$ if x is odd

Hence, it is primitive recursive.

Example 11.6 `Prove that the function fact(x) is primitive recursive.`