§1. Primitive Recursive Functions: In this chapter, it will be helpful if we think of a function as a rule which assigns an output to given input. The basic idea is to express the output in terms of the input. A function is still mathematically a set of ordered pairs which satisfies the condition \((x, y) \in f \land (x, z) \in f \Rightarrow y = z\). An \(n\)-ary function on \(\mathbb{N}\) is just a function from \(\mathbb{N}^n\) to \(\mathbb{N}\). When \(n = 0\), \(\mathbb{N}^0 = \) set of 0-tuples = \{\emptyset\}.

\[
\begin{align*}
\xrightarrow{\text{f}} & \quad \xrightarrow{\text{g}} & \quad \xrightarrow{\text{h}} \\
\text{function of} & \quad \text{function of} & \quad \text{function of} \\
1 \text{ variable} & \quad 2 \text{ variables} & \quad 0 \text{ variables} \\
f: \mathbb{N} \rightarrow \mathbb{N} & \quad f: \mathbb{N}^2 \rightarrow \mathbb{N} & \quad f: \mathbb{N}^0 \rightarrow \mathbb{N}.
\end{align*}
\]

A constant can be viewed as a function of 0 variables. 

Def. A function \(f: \mathbb{N}^k \rightarrow \mathbb{N}\) is said to be primitive recursive if it can be obtained from the initial functions by a finite number of applications of cartesian products, compositions, & primitive recursions.

Def. The initial functions are:
(a) the zero function of \(0\) variables: \(0\)
(b) the zero function of 1 variable: \(z(x) = 0\) for all \(x \in \mathbb{N}\)
(c) the successor function: \(s(x) = \text{successor of} x, \text{for all} x \in \mathbb{N}\)
(d) the projective functions: \(I_k, n (x_1, \ldots, x_n) = x_k\) for \(1 \leq k \leq n\); and \(I_{n+1} (x_1, \ldots, x_n) = \lambda\).
Example 1: Here are some values of some initial functions.

(a) \( Z(2) = 0, \ Z(5) = 0, \ Z(8) = 0 \)
(b) \( S(2) = 3, \ S(5) = 6, \ S(8) = 9 \)
(c) \( I_{1,2}(5, 7) = 5, \ I_{2,3}(3, 4, 5) = 4, \ I_{1,1}(4) = 4, \ I_{2,2}(4, 6) = 6, \ I_{0,2}(3, 4) = 7, \ I_{0,1}(7) = 7 \)

**Def.** Let \( g : \mathbb{N}^n \rightarrow \mathbb{N}^k \) and \( h : \mathbb{N}^m \rightarrow \mathbb{N}^m \) be functions. The cartesian product of the functions \( g \) & \( h \) is the function \( f : \mathbb{N}^{n+m} \rightarrow \mathbb{N}^{n+m} \) defined by
\[
f(x) = \langle g(x), h(x) \rangle \quad \& \quad \text{we write } f = g \times h
\]
Here \( x \) abbreviates \( \langle x_1, \ldots, x_n \rangle \).

**Example 2:** Let \( f = S \times I_{1,1} \). Then \( f : \mathbb{N} \rightarrow \mathbb{N}^2 \)
is given by \( f(x) = \langle S(x), I_{1,1}(x) \rangle = \langle x+1, x \rangle \)

**Def.** Let \( h_1, h_2, \ldots, h_k : \mathbb{N}^n \rightarrow \mathbb{N} \) and \( g : \mathbb{N}^2 \rightarrow \mathbb{N} \) be functions. Then we can define a new function
\( f : \mathbb{N}^n \rightarrow \mathbb{N} \) by putting
\[
f(x) = g(h_1(x), h_2(x), \ldots, h_k(x))
\]
Here \( x \) again abbreviates \( \langle x_1, \ldots, x_n \rangle \). The function \( f \) is said to be obtained from \( g \) and \( h_1, h_2, \ldots, h_k \) by composition & we write \( f = g \circ (h_1 \times h_2 \times \ldots \times h_n) \)

**Example 3** Let \( g : \mathbb{N}^3 \rightarrow \mathbb{N} \) be defined by
\[
g(x_1, y_1, z_1) = 2x_1 + 3y_1 + 3z_1^2
\]
Also let \( h_1 : \mathbb{N}^2 \rightarrow \mathbb{N} \)
be defined by \( h_1(x_1, x_2) = x_1 + x_2 \), \( h_2(x, y) = x \cdot y \)
and \( h_3(x_1, x_2) = x_1 - x_2 \). Then if \( f = g \circ (h_1 \times h_2 \times h_3) \)
we will have
\[
f(x_1, x_2) = g(h_1(x_1, x_2), h_2(x_1, x_2), h_3(x_1, x_2)) = 2(x_1 + x_2) + 3x_1x_2 + (8x_1)
\]
Def. Let \( g : \mathbb{N}^n \to \mathbb{N} \) & \( h : \mathbb{N}^{n+2} \to \mathbb{N} \) be functions where \( n \geq 0 \).

Then we can define a new function \( f : \mathbb{N}^{n+1} \to \mathbb{N} \) by putting
\[
 f(x, 0) = g(x) \quad \text{&} \quad f(x, s(y)) = h(x, y, f(x, y)).
\]
Here \( x \) again abbreviates \( \langle x_1, \ldots, x_n \rangle \). The function \( f \) is said to be obtained from \( g \) and \( h \) by primitive recursion and we write \( f = \text{prec}(g, h) \).

Ex. 4 Let \( g : \mathbb{N} \to \mathbb{N} \) be defined by \( g(x) = x \) and let \( h : \mathbb{N}^3 \to \mathbb{N} \) be defined by \( h(x_1, x_2, x_3) = 2x_3 + x_2 + 1 \).

Now if \( f = \text{prec}(g, h) \), then \( f : \mathbb{N}^2 \to \mathbb{N} \) and
\[
 f(x, 0) = g(x) = x,
 f(x, 1) = f(x, s(0)) = h(x, 0, f(x, 0)) = 2f(x, 0) + 0 + 1 = 2x + 1,
 f(x, 2) = f(x, s(1)) = h(x, 1, f(x, 1)) = 2f(x, 1) + 1 + 1 = 2(2x + 1) + 1 + 1 = 4x + 4,
 f(x, 3) = f(x, s(2)) = h(x, 2, f(x, 2)) = 2f(x, 2) + 2 + 1 = 2(4x + 4) + 2 + 1 = 8x + 11,
 f(x, 4) = f(x, s(3)) = h(x, 3, f(x, 3)) = 2f(x, 3) + 3 + 1 = 2(8x + 11) + 3 + 1 = 16x + 26.
\]
In general
\[
 f(x, y) = 2^y x + 2^{y+1} - (y + 2), \quad \text{but it is not easy to see this}.
\]

Ex. 5 Let \( g = \) the constant 1 and \( h(x_1, x_2) = 2x_2 + 1 \). Then \( g : \mathbb{N}^0 \to \mathbb{N} \) & \( h : \mathbb{N}^2 \to \mathbb{N} \). So if we put \( f = \text{prec}(g, h) \) we will get a function \( f : \mathbb{N} \to \mathbb{N} \).
Now \( f(0) = g = 1 \)

\[
\begin{align*}
\text{Ex. 5} & \\
f(1) &= f(s(0)) = h(0, f(0)) \\
&= 2 \cdot f(0) + 1 = 2 + 1 = 3 \\
f(2) &= f(s(1)) = h(1, f(1)) \\
&= 2 \cdot f(1) + 1 = 2(3) + 1 = 7 \\
f(3) &= f(s(2)) = h(2, f(2)) \\
&= 2 \cdot f(2) + 1 = 2(7) + 1 = 15.
\end{align*}
\]

In general, it is very easy to see that \( f(y) = 2^{y+1} - 1 \).

\[
\text{Ex. 6} \\
\text{Now let } g = 1 \text{ and } h(x_1, x_2) = 2x_2 + x_1.
\]

Then \( g: \mathbb{N}^0 \to \mathbb{N} \) and \( h: \mathbb{N}^2 \to \mathbb{N} \). So if \( f = \text{prece}(g, h) \) then \( f \) will be a function from \( \mathbb{N}^1 \) to \( \mathbb{N} \). Also

\[
\begin{align*}
\text{Ex. 6} & \\
f(0) &= g = 1 \\
f(1) &= f(s(0)) = h(0, f(0)) \\
&= 2 \cdot f(0) + 0 = 2 \\
f(2) &= f(s(1)) = h(1, f(1)) \\
&= 2 \cdot f(1) + 1 = 2(2) + 1 = 5 \\
f(3) &= f(s(2)) = h(2, f(2)) \\
&= 2 \cdot f(2) + 2 = 2(5) + 2 = 12 \\
f(4) &= f(s(3)) = h(3, f(3)) \\
&= 2 \cdot f(3) + 3 = 2(12) + 3 = 27 \\
f(5) &= f(s(4)) = h(4, f(4)) \\
&= 2 \cdot f(4) + 4 = 2(27) + 4 = 58 \\
f(6) &= f(s(5)) = h(5, f(5)) \\
&= 2 \cdot f(5) + 5 = 2(58) + 5 = 121
\end{align*}
\]

In general, \( f(y) = 2^{y+1} - (y+1) \) but this is moderately difficult to see.
Demonstrating that certain functions are primitive recursive.

**Ex. 1** Let $f(x) = 3$ for each $x \in \mathbb{N}$. Show that $f : \mathbb{N} \to \mathbb{N}$ is a primitive recursive function.

**Sol.** $f(x) = s(s(s(z(x))))$. So $f = s \circ s \circ s \circ z$. Hence $f$ can be obtained from the initial functions by a finite number of applications of compositions. So $f$ is primitive recursive.

**Ex. 2** Let $g(x,y) = y+2$. Show that $g : \mathbb{N}^2 \to \mathbb{N}$ is primitive recursive.

**Sol.** $g(x,y) = s(s(1_{2,2}(x,y)))$. So $g = s \circ s \circ 1_{2,2}$. Hence $g$ is a primitive recursive function.

**Ex. 3** Let $ADD(x,y) = x + y$. Show that $ADD : \mathbb{N}^2 \to \mathbb{N}$ is a primitive recursive function.

**Sol.** We will find primitive recursive functions $g$ and $h$ such that $ADD = \text{prec}(g, h)$. Now

$ADD(x,0) = x \leftarrow g(x)$

$ADD(x, s(y)) = ADD(x, y+1) = (x+y)+1 = ADD(x, y)+1$

$= s(ADD(x, y)) \leftarrow h(x, y, ADD(x, y))$

Hence $g(x) = x$. So $g = 1_{1,1}$.

And $h(x, y, ADD(x, y)) = s(ADD(x, y))$. So $h = s \circ 1_{3,3}$. Thus $ADD = \text{prec}(g, h) = \text{prec}(1_{1,1}, s \circ 1_{3,3})$. Hence $ADD$ is primitive recursive.
Ex. 4 Let $\text{MULT}(x, y) = x \cdot y$. Show that $\text{MULT} : \mathbb{N} \to \mathbb{N}$ is a primitive recursive function.

Sol. Again we will find primitive recursive functions $g$ and $h$ such that $\text{MULT} = \text{prec}(g, h)$. Now

$$\text{MULT}(x, 0) = x \cdot 0 = 0 \leftarrow g(x)$$

$$\text{MULT}(x, s(y)) = \text{MULT}(x, y + 1) = x \cdot (y + 1)$$

$$= x \cdot y + x = \text{MULT}(x, y) + x$$

$$= \text{ADD}(\text{MULT}(x, y), x) \leftarrow h(x, y, \text{MULT}(x, y))$$

\[ g(x) = 0 \text{ for each } x \in \mathbb{N}. \] So $g = \varepsilon$. And

$$h(x, y, \text{MULT}(x, y)) = \text{ADD}(\text{MULT}(x, y), x).$$ So

$$h = \text{ADD} \circ (I_{3,3} \land I_{1,3}) \circ (I_{3,3} \land I_{1,3}).$$ Hence $\text{MULT} = \text{prec}(g, h)$

$$= \text{prec}(\varepsilon, \text{ADD} \circ (I_{3,3} \land I_{1,3})$$

$$= \text{prec}(\varepsilon, \text{prec}(I_{1,1} \land s(I_{3,3} \circ (I_{3,3} \land I_{1,3})))$$

Thus $\text{MULT}$ is primitive recursive.

Ex. 5 Let $\text{PRED}(y) = \begin{cases} 0 & \text{if } y = 0, \\ y - 1 & \text{if } y > 0. \end{cases}$ Show that $\text{PRED} : \mathbb{N} \to \mathbb{N}$ is primitive recursive.

Sol. We will find primitive recursive functions $g$ and $h$ such that $\text{PRED} = \text{prec}(g, h)$. Now

$$\text{PRED}(0) = 0 \leftarrow g$$

$$\text{PRED}(s(y)) = s(y) - 1 = y \leftarrow h(y, \text{PRED}(y))$$

So $g = 0$ and $h(y, \text{PRED}(y)) = y$. Thus

$$h = I_{1,2}. \text{ Hence}$$

$$\text{PRED} = \text{prec}(g, h)$$

$$= \text{prec}(0, I_{1,2}).$$

Thus $\text{PRED}$ is a primitive recursive function.
Ex. 6. Let \( \text{MONUS}(x, y) = \begin{cases} x - y & \text{if } x \geq y, \\ 0 & \text{if } x < y \end{cases} \). We usually write \( x - y \) for \( \text{MONUS}(x, y) \). Show that \( \text{MONUS} \) is a primitive recursive function.

\[ \text{MONUS}(x, 0) = x - 0 = x \iff g(x) \]
\[ \text{MONUS}(x, s(y)) = x - s(y) = x - (y + 1) = \text{PRED}(x - y) \iff h(x, y, \text{MONUS}(y)) \]

So \( g(x) = x \) for each \( x \in \mathbb{N} \). Thus \( g = I_{1, 1} \). Also \( h(x, y, \text{MONUS}(y)) = \text{PRED}(\text{MONUS}(x, y)) \). Hence
\[ \text{MONUS} = \text{prec}(g, h) = \text{prec}(I_{1, 1}, \text{PRED} \circ I_{3, 3}) \]
\[ = \text{prec}(I_{1, 1}, \text{prec}(0, I_{1, 2}) \circ I_{3, 3}) \]
Thus \( \text{MONUS} \) is a primitive recursive function.

Ex. 8. Let \( \text{SWITCH}(x, y) = \langle y, x \rangle \). Show that \( \text{SWITCH} : \mathbb{N}^2 \to \mathbb{N}^2 \) is a primitive recursive function.

\[ \text{SWITCH}(x, y) = \langle I_{2, 2}(x, y), I_{1, 2}(x, y) \rangle \]
So \( \text{SWITCH} = I_{2, 2} \wedge I_{1, 2} \).

Ex. 9. Let \( \text{SIGN}(x) = \begin{cases} 0 & \text{if } x = 0, \\ 1 & \text{if } x > 0, \\ 0 & \text{if } x < 0 \end{cases} \). Show that \( \text{SIGN} \) and \( \overline{\text{SIGN}} \) are primitive recursive.

\[ \text{SIGN}(0) = 0 \land \text{SIGN}(s(y)) = 1 \]. So \( \text{SIGN} = \text{prec}(0, s \circ 0 I_{1, 2}) \)
\[ \overline{\text{SIGN}}(0) = 1 \land \overline{\text{SIGN}}(s(y)) = 0 \]. So \( \overline{\text{SIGN}} = \text{prec}(s \circ 0, z \circ I_{1, 2}) \).
§3  Ackermann's function.

Most of the "natural" functions that we encounter in Number Theory are primitive recursive. It is in fact quite difficult to find a function from \( \mathbb{N} \) to \( \mathbb{N} \) which is not primitive recursive.

**Def.** The Ackermann's function \( A : \mathbb{N}^2 \to \mathbb{N} \) is defined by simultaneous recursion as follows.

(a) \( A(x+1, y+1) = A(x, A(x+1, y)) \),
(b) \( A(x+1, 0) = A(x, 1) \), and
(c) \( A(0, y) = y + 1 \).

**Theorem 1.** Let \( f(x) = A(x, x) \). Then \( f: \mathbb{N} \to \mathbb{N} \) is not a primitive recursive function.

The essence of the proof is to show that \( f(x) \) grows faster than each primitive recursive function, but it is way too complicated to include in these notes.

Let us calculate \( A(x, y) \) for various small values of \( x \) and \( y \).

**Ex. 1.**

\[
\begin{align*}
A(0, y) &= y + 1 \quad \text{by (c)} \\
A(1, 0) &= A(0+1, 0) = A(0, 1) = 2 \quad \text{by (b)} \\
A(1, 1) &= A(0+1, 0+1) = A(0, A(0, 1)) \quad \text{by (a)} \\
&= A(0, A(1, 0)) = A(0, 2) = 3 \\
A(1, 2) &= A(0+1, 1+1) = A(0, A(0, 1)) \quad \text{by (a)} \\
&= A(0, A(1, 1)) = A(0, 3) = 4 \\
\end{align*}
\]
Ex. 2 In general, it can be shown that \( A(0, y) = y + 2 \).

Now \( A(2, 0) = A(1+1, 0) = A(1, 1) = 3 \) by (6).

\[
A(2, 1) = A(1+1, 0+1) = A(1, A(1, 0)) \quad \text{by (a)}
\]
\[
= A(1, A(0)) = A(1, 1) = 5
\]

\[
A(2, 2) = A(1+1, 1+1) = A(1, A(1+1, 1)) \quad \text{by (a)}
\]
\[
= A(1, A(2)) = A(1, 5) = 7
\]

\[
A(2, 3) = A(1+1, 2+1) = A(1, A(1+1, 2)) \quad \text{by (a)}
\]
\[
= A(1, A(2)) = A(1, 7) = 9.
\]

In general, it can be shown that \( A(2, y) = 2y+3 \).

Ex. 3 \[
A(3, 0) = A(2+1, 0) = A(2, 1) \quad \text{by (6)}
\]
\[
= 5 = 2^3 - 3
\]

\[
A(3, 1) = A(2+1, 0+1) = A(2, A(2+1, 0)) \quad \text{by (a)}
\]
\[
= A(2, A(0)) = A(2, 5) = 2(2^3 - 3) + 3 = 2^{1+3} - 3
\]

\[
A(3, 2) = A(2+1, 1+1) = A(2, A(2+1, 1)) \quad \text{by (a)}
\]
\[
= A(2, A(3)) = A(2, 2^{1+3} - 3)
\]
\[
= 2(2^{1+3} - 3) + 3 = 2^{2+3} - 3.
\]

In general, it can be shown that \( A(3, y) = 2^{y+3} - 3 \).

Ex. 4 \[
A(4, 0) = A(3+1, 0) = A(3, 1) \quad \text{by (6)}
\]
\[
= 2^4 - 3 = 2^2 - 3
\]

\[
A(4, 1) = A(3+1, 0+1) = A(3, A(3+1, 0)) \quad \text{by (a)}
\]
\[
= A(3, 2^4 - 3) = 2^{2^4 - 3 + 3} - 3 = 2^{2^4} - 3
\]

\[
A(4, 2) = A(3+1, 1+1) = A(3, A(3+1, 1)) \quad \text{by (a)}
\]
\[
= A(3, 2^4 - 3) = 2^{2^4 - 3 + 3} - 3 = 2^{2^4} - 3.
\]

In general, it can be shown that \( (y+3)2^y - 3 \).

\[
A(4, y) = \begin{cases} 2^y - 3 & \text{for } y \geq 3 \\ 2 & \text{for } y = 0, 1, 2 \end{cases}
\]
Theorem 2: For each \( x \in \mathbb{N} \), we have

(i) \( A(1, y) = y + 2 \)

(ii) \( A(2, y) = 2(y+3) - 3 \)

(iii) \( A(3, y) = 2^{y+3} - 3 \)

(iv) \( A(4, y) = \frac{(y+3)^2}{2^5} \cdot 2^2 - 3 \)

Proof: The proofs are all by induction on \( y \).

(i) For \( y = 0 \), we have \( A(1, 0) = A(0, 1) = 2 = 0 + 2 \).
So the result is true for \( y = 0 \). Suppose the result is true for \( y = 0 \). Then \( A(1, y) = y + 2 \).

\[ A(1, y+1) = A(0, A(1, y)) = A(0, y+2) \]
by induction.

So if the result is true for \( y \), it will be true for \( y + 1 \).
Hence the result is true for all \( y \).

(ii) \( A(2, 0) = A(1, 1) = 3 = 2(0+3) - 3 \).
So result is true for \( y = 0 \).
Now suppose the result is true for \( y = 0 \). Then \( A(2, y) = 2(y+3) - 3 \).

\[ A(2, y+1) = A(1, A(2, y)) = A(1, 2(y+3) - 3) \]
by induction.

So if the result is true for \( y \), it will be true for \( y + 1 \).
Hence the result is true for all \( y \).

(iii) \( A(3, 0) = A(2, 1) = 5 = 2^{0+3} - 3 \).
So the result is true for \( y = 0 \).
Suppose the result is true for \( y \). Then \( A(3, y) = 2^{y+3} - 3 \).

\[ A(3, y+1) = A(2, A(3, y)) = A(2, 2^{y+3} - 3) \]
by induction.

So if the result is true for \( y \), it will be true for \( y + 1 \).
Hence the result is true for all \( y \) by induction.

(iv) \( A(4, 0) = A(3, 1) = 4^2 - 3 = 2^2 - 3 \).
So result is true for \( y = 0 \).
Suppose result is true for \( y \). Then \( A(4, y) = 2^{y+3} - 3 \).

\[ A(4, y+1) = A(3, A(4, y)) = A(3, 2^{y+3} - 3) \]
with \((y+1)+3\) 2's.
So result is true for all \( y \).
Recursive functions.

We can define a wider class of functions than the primitive recursive ones by using simultaneous recursion instead of primitive recursion — but all the functions we obtain will be total functions. In order to also obtain partial functions, we will introduce a new operation called minimization and this will lead us to the same wider class of total functions.

**Def.** Let \( g : \mathbb{N}^{n+1} \rightarrow \mathbb{N} \) be a total function and \( n \geq 0 \). Then we can define a partial function \( f : \mathbb{N}^n \rightarrow \mathbb{N} \) by putting

\[
f(x) = \begin{cases} \text{smallest value of } y \text{ such that } g(x, y) = 0 \\ \text{undefined} \quad \text{if } g(x, y) > 0 \text{ for each } y \in \mathbb{N}. \end{cases}
\]

Here \( x \) abbreviates \( \langle x_1, \ldots, x_n \rangle \) as usual. The function \( f \) is said to be obtained from \( g \) by minimization and we write \( f = \mu[g, 0] \). We also sometimes write \( f(x) = (\mu y)[g(x, y) = 0] \).

**Ex. 1.** Let \( g(x, y) = x - 3y \). Then \( g : \mathbb{N}^2 \rightarrow \mathbb{N} \) is a total function. If \( f = \mu(g, 0) \), then

\[
\begin{align*}
f(4) &= (\mu y)[4 - 3y = 0] = 2 \\
f(8) &= (\mu y)[8 - 3y = 0] = 3 \\
f(9) &= (\mu y)[9 - 3y = 0] = 3 \\
f(25) &= (\mu y)[25 - 3y = 0] = 9 \\
f(30) &= (\mu y)[30 - 3y = 0] = 10.
\end{align*}
\]

In general \( f(x) = (\mu y)[x - 3y = 0] = \lceil x/3 \rceil \) where \( \lceil z \rceil = \text{smallest integer } \geq z \) = ceiling function of \( z \).
Ex. 2. Let \( g(x,y) = (x - 3y) + (3y - x) \). Then \( g(x,y) = |x - 3y| \).

If \( f = \mu[9,0] \), then

\[ f(0) = (\mu y)[|10 - 3y| = 0] = 0 \]
\[ f(1) = (\mu y)[|11 - 3y| = 0] = \text{undefined} \]
\[ f(2) = (\mu y)[|12 - 3y| = 0] = \text{undefined} \]
\[ f(3) = (\mu y)[|13 - 3y| = 0] = 1 \]

In general, \( f(x) = \begin{cases} \frac{x}{3}, & \text{if } x \text{ is a multiple of } 3 \\ \text{undefined}, & \text{if } x \text{ is not a multiple of } 3 \end{cases} \)

Def. A partial function \( f : \mathbb{N}^k \rightarrow \mathbb{N} \) is said to be recursive (or \( \mu \)-recursive) if it can be obtained from the initial functions by a finite number of applications of cartesian products, compositions, primitive recursions and minimization on total functions.

Ex. 1. Let \( f(x) = \lceil \frac{x}{2} \rceil = \text{smallest integer } \leq \frac{x}{2} \). Show that \( f \) is a recursive function.

Sol. Let \( g(x,y) = x \div 2y \). Then \( g : \mathbb{N}^2 \rightarrow \mathbb{N} \) is a total function and if we put \( f = \mu[9,0] \), then

\[ f(x) = (\mu y)[g(x,y) = 0] = (\mu y)[\frac{x}{2y} = 0] \]

\[ = \text{smallest } y \text{ such that } (x \div 2y = 0) = \lfloor \frac{x}{2} \rfloor \]

Now \( g(x,y) = \text{MONUS}(x, 2y) = \text{MONUS}(x, y+y) \]
\[ = \text{MONUS}(I_{1,2}(x,y), \text{ADD}(I_{2,2}(x,y), I_{2,2}(x,y))) \]

So \( f = \mu[9,0] \)
\[ = \mu[\text{MONUS} \circ (I_{1,2} \land \text{ADD} \circ (I_{2,2} \land I_{2,2})), 0] \]

Hence \( f \) is a recursive function because it is obtained from the initial functions using the 4 operations.
Ex. 2 Let \( f_2(x) = \sqrt[3]{2x} \). Show that \( f_2 \) is a recursive function.

**Sol.** Let \( g(x, y) = 2x - 3y \). Then \( g \) is a total function.
Also \( (\forall y)[g(x, y) = 0] = (\forall y)[2x - 3y = 0] \)
\[ = \text{smallest } y \text{ such that } (2x - 3y = 0) \]
\[ = \sqrt[3]{2x} = f_2(x). \]

Now \( g(x, y) = \text{MONUS}(\text{MULT}(2, x), \text{MULT}(3, y)) \)
\[ = \text{MONUS}(\text{MULT}(2, x), \text{MULT}(3, y)). \]

\[ \therefore f_2 = \mu[g, 0] \]
\[ = \mu[\text{MONUS}(\text{MULT}(2, x), \text{MULT}(3, y)).] \]

Ex. 3 Let \( f_3(x) = \sqrt{x^3} \). Show that \( f_3 \) is a recursive function.

**Sol.** Let \( g(x, y) = x - y^2 \). Then \( g \) is a total function. Also
\( (\forall y)[g(x, y) = 0] = (\forall y)[x - y^2 = 0] \)
\[ = \text{smallest } y \text{ such that } (x - y^2 = 0) \]
\[ = \sqrt[3]{x} \]

Now \( g(x, y) = \text{MONUS}(x, \text{MULT}(y, y)) \)
\[ = \text{MONUS}(x, \text{MULT}(y, y)). \]

\[ \therefore g = \text{MONUS}(I_{1, 2}(x, y), \text{MULT}(y, y)) \]
\[ = \mu[g, 0] \]
\[ = \mu[\text{MONUS}(I_{1, 2}(x, y), \text{MULT}(y, y))]. \]

Ex. 4 Let \( f_4(x) = \begin{cases} x/2 & \text{if } x \text{ is an even integer} \\ \text{undefined} & \text{if } x \text{ is an odd integer} \end{cases} \)

Show that \( f_4 \) is a recursive partial function.

**Sol.** Let \( g(x, y) = (x - 2y) + (2y - x) \). Then \( g \) is a total function and
\( (\forall y)[g(x, y) = 0] = (\forall y)[x - 2y = 0] \)
Ex 4. The smallest $y$ such that $\text{ABS}(x, 2y) = 0$ is $f(x)$ because:

- If $x$ is even, $x/2$ is the only integer for which $|x-2y| = 0$.
- And when $x$ is odd, $|x-2y| > 1$ for all $y$.

Now $g(x, y) = \text{ABS}(x, 2y) = \text{ABS}(I_{1,2}(x, y), \text{ADD}(I_{1,2}(x, y), I_{2,2}(x, y)))$

Thus $f = \mu [g, 0]$

$= \mu [\text{ABS} \circ (I_{1,2} \land \text{ADD} \circ (I_{2,2} \land I_{1,2})), 0]$

Hence $f$ is a recursive partial function because $\text{ABS}$ & $\text{ADD}$ are primitive recursive total functions.

ABS$ = \text{ADD} \circ (\text{MONUS} \land \text{MONUS} \circ (I_{2,2} \land I_{1,2}))$.

Ex 5. Let $f_5(x) = \begin{cases} x^{1/3} & \text{if } x \text{ is a perfect cube} \\ \text{undefined} & \text{if } x \text{ is not a perfect cube} \end{cases}$

Show that $f_5$ is recursive.

Sol. Let $g(x, y) = (x-y^3) = \text{ABS}(x, y^3)$. Then it is not difficult to see that $(\mu y)[g(x, y) = 0] = f_5(x)$. So

$f_5 = \mu [g, 0]$

$= \mu [\text{ABS} \circ (I_{1,2} \land \text{MULT} \circ (\text{MULT} \circ (I_{2,2} \land I_{1,2}) \land I_{2,2})), 0]$.

Hence $f_5$ is recursive.

Ex 6. Let $h_1(x) = \begin{cases} 0 & \text{if } x \text{ is even} \\ 1 & \text{if } x \text{ is odd} \end{cases}$

Show that $f$ is recursive.

Sol. $h_1(x) = \begin{cases} \lceil x/2 \rceil & \text{if } x \text{ is odd} \\ x & \text{otherwise} \end{cases}$

Then $f_1(x) = \lfloor x/2 \rfloor$. Then

$h_1 = \text{MONUS} \circ (\text{MULT} \circ (S \circ S \circ 0 \circ I_{1,1}) \land f_1) \land I_{1,1}$

is a recursive function because $f_1$ is a recursive function from Ex 1.
Ex. 7. Let \( h_2(x) = \begin{cases} 0 & \text{if } x \text{ is a perfect square} \\ 1 & \text{if } x \text{ is not a perfect square} \end{cases} \)

Also let \( h_3(x) = \begin{cases} 1 & \text{if } x \text{ is a perfect square} \\ 0 & \text{if } x \text{ is not a perfect square} \end{cases} \)

Show that \( h_2 \) & \( h_3 \) are recursive functions.

Sol.

\[
\begin{align*}
    h_2(x) &= \text{SIGN}\left(\frac{\sqrt{x-1}}{x}\right) \\
    &= \text{SIGN}\left(\text{MONUS}\left(\text{MULT}\left(f_3(x), f_3(x)\right), I_{11}(x)\right)\right), \\
    \text{where } f_3 \text{ is the recursive function from Example 3, } \\
    \text{So } h_2 &= \text{SIGN} \circ \text{MONUS} \circ (\text{MULT} \circ (f_3 \land f_3) \land I_{11}) \text{ is a recursive function.} \\
    h_3(x) &= 1 - h_2(x). \text{ So } h_3 = \text{MONUS} \circ (\text{SOE} \land I_{11} \land h_2) \text{ is a recursive function.}
\end{align*}
\]

Ex. 8. Let \( h_4(x) = \lfloor \sqrt{x} \rfloor \) be the largest integer \( \leq \sqrt{x} \). Show that \( h_4 \) is a recursive function.

Sol.

Recall that \[
\lfloor z \rfloor = \begin{cases} \frac{z}{7} & \text{if } z \in \mathbb{N}, \\
\frac{z}{7} + 1 & \text{if } z \notin \mathbb{N}.
\end{cases}
\]

So \( h_4(x) = \lfloor \sqrt{x} \rfloor = \begin{cases} \frac{\sqrt{x}}{7} & \text{if } \sqrt{x} \in \mathbb{N}, \\
\frac{\sqrt{x}}{7} + 1 & \text{if } \sqrt{x} \notin \mathbb{N}.
\end{cases}
\]

\[
\begin{align*}
    &= \frac{\sqrt{x}}{7} - h_2(x) \quad \text{where } h_2 \text{ is from Ex. 7} \\
    &= f_3(x) - h_2(x) \quad \text{where } f_3 \text{ is from Ex. 3} \\
    \therefore h_4 &= \text{MONUS} \circ (f_3 \land h_2) \text{ and so is a recursive function.}
\end{align*}
\]

Remember \( \text{ABS}(x, y) = (x \equiv y) \lor (y \equiv x) \) & \( \text{SIGN}(x) = \begin{cases} 0 & \text{if } x = 0, \\
1 & \text{if } x > 0.
\end{cases} \)
§5. Recursive & Semi-recursive relations.

Def. Let $R \subseteq \mathbb{N}^n$ be an $n$-ary relation on $\mathbb{N}$. We define the characteristic function $\chi_R$ of $R$ by

$$\chi_R(x) = \begin{cases} 1 & \text{if } x \in R, \\ 0 & \text{if } x \notin R. \end{cases}$$

Here $x = (x_1, \ldots, x_n)$.

We define the affirmative function $\alpha_R$ of $R$ by

$$\alpha_R(x) = \begin{cases} 1 & \text{if } x \in R, \\ \text{undefined} & \text{if } x \notin R. \end{cases}$$

Def. The $n$-ary relation $R \subseteq \mathbb{N}^n$ is a recursive relation if its characteristic function $\chi_R$ is a recursive function. $R$ is said to be a semi-recursive relation if its affirmative function, $\alpha_R$, is a recursive function.

Ex.1 Let $R$ be the $1$-ary relation defined by $x \in R$ if $x$ is odd. Then

$$\chi_R(x) = \begin{cases} 1 & \text{if } x \text{ is odd}, \\ 0 & \text{if } x \text{ is not odd}. \end{cases}$$

Then $\chi_R$ is a recursive function because $\chi_R = \lambda_x$, from Ex. 6 of the previous section. So $R$ is a recursive relation.

Ex.2 Let $R$ be the binary relation defined by $(x_1, x_2) \in R$ if $x_1 > x_2$. Then

$$\chi_R(x_1, x_2) = \begin{cases} 1 & \text{if } x_1 > x_2, \\ 0 & \text{if } x_1 \leq x_2. \end{cases}$$

So $\chi_R = \text{SIGN} \circ \text{MONUS}$ is a recursive function. Hence $R$ is a recursive relation.
Now it can be shown that a function \( f \) is recursive if and only if we can find a recursive total function \( g \) such that \( f = \mu[g, 0] \). Since \( \alpha_A \) and \( \alpha_B \) are recursive functions, this means that we can find recursive total functions \( g_A \) and \( g_B \) such that \( \mu[g_A, 0] = \alpha_A \) and \( \mu[g_B, 0] = \alpha_B \).

(a) \[
\alpha_{A \land B}(x) = \begin{cases} 
1 & \text{if } x \in A \land B \\
\text{undefined} & \text{if } x \notin A \land B
\end{cases}
\]
\[
= (\mu y) [ g_A(x, y) + g_B(x, y) = 0 ]
\]
\[
\therefore \alpha_{A \land B} = \mu [ \text{ADD} \circ (g_A \lor g_B), 0 ] . \text{ Hence } \alpha_{A \land B} \text{ is a recursive function. So } A \land B \text{ is a semi-recursive n-ary relation.}
\]

(b) Also \[
\alpha_{A \lor B}(x) = \begin{cases} 
1 & \text{if } x \in A \lor B \\
\text{undefined} & \text{if } x \notin A \lor B
\end{cases}
\]
\[
= (\mu y) [ g_A(x, y) \lor g_B(x, y) = 0 ]
\]
\[
\therefore \alpha_{A \lor B} = \mu [ \text{MULT} \circ (g_A \lor g_B), 0 ] . \text{ Hence } \alpha_{A \lor B} \text{ is a recursive function. So } A \lor B \text{ is a semi-recursive n-ary relation.}
\]

**Note** If \( A \) is a semi-recursive n-ary relation, it does not always follow that \( A^c \) is a semi-recursive n-ary relation. For example, let \( R \) be the binary relation on \( \mathbb{N} \) defined by
\[
c(M) RC(w) \iff \text{the TM } M \text{ halts on the input } w.
\]
Here \( c(M) \) \& \( q(w) \) are codings of \( M \) \& \( w \) into natural numbers. Then \( R \) is a semi-recursive relation which is not recursive. This forces \( R^c \) to be a non-semi-recursive relation.
Proposition 3: If $A \& B$ are $n$-ary recursive relations, then so are (a) $A^c$ (b) $A \cap B$ (c) $A \cup B$.

Proof: Suppose $A \& B$ are recursive relations. Then $\chi_A$ and $\chi_B$ will be recursive functions.

(a) Now $\chi_{A^c}(x) = \begin{cases} 1 & \text{if } x \in A^c \\ 0 & \text{if } x \notin A^c \end{cases}$

So $\chi_{A^c}(x) = 1 - \chi_A(x)$. Hence $\chi_{A^c} = \text{MONUS} \cdot (\text{SOZ} \circ I_{1,n} \land \chi_A)$ is a recursive function. So $A^c$ is a recursive relation.

(b) Also $\chi_{A \cap B} = \begin{cases} 1 & \text{if } x \in A \cap B \\ 0 & \text{if } x \notin A \cap B \end{cases} = \chi_A(x) \cdot \chi_B(x)$

So $\chi_{A \cap B} = \text{MUL} \cdot (\chi_A \land \chi_B)$ is a recursive function. Hence $A \cap B$ is a recursive relation.

(c) $\chi_{A \cup B}(x) = \begin{cases} 1 & \text{if } x \in A \cup B \\ 0 & \text{if } x \notin A \cup B \end{cases} = 1 - \text{MUL} \left(1 - \chi_A(x), 1 - \chi_B(x)\right)$.

So $\chi_{A \cup B} = \text{MONUS} \circ (\text{SOZ} \circ I_{1,n} \land \text{MUL} \circ \text{MONUS} \circ (\text{SOZ} \circ I_{1,n} \land \chi_A) \land \text{MONUS} \circ (\text{SOZ} \circ I_{1,n} \land \chi_B))$ is a recursive function. Hence $A \cup B$ is a recursive relation.

Proposition 4. If $A \& B$ are $n$-ary semi-recursive relations, then so are (a) $A \cap B$ (b) $A \cup B$.

Proof: Suppose $A \& B$ are semi-recursive relations. Then $\chi_A$ and $\chi_B$ will be recursive functions.
Proposition 5. If $A$ and $A^c$ are both semi-recursive $n$-ary relations, then $A$ must be a recursive relation.

Proof: Suppose $A$ and $A^c$ are semi-recursive relations. Then $\chi_A$ and $\chi_{A^c}$ will be recursive partial functions. So as in the proof of Prop 4, we can find recursive total functions $g$ and $h$ such that $\chi_A = \mu [g, 0]$ and $\chi_{A^c} = \mu [h, 0]$.

Now $\chi_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \in A^c \end{cases}
= \{(\mu y) \mid g(x, y) = 0 \} \text{ if } x \in A
= \{(\mu y) \mid h(x, y) = 0 \} \text{ otherwise}
= \{(\mu y) \mid g(x, y) = 0 \} - 1 \text{ if } x \in A
= \{(\mu y) \mid h(x, y+1) = 0 \} \text{ if } x \in A^c
= (\mu y) \left[ \min \{g(x, y), h(x, y+1)\} = 0 \right]

But $\min(x, y) = x \div (x \times y)$. So $\min = \text{monus} \circ (I_{1,2} \land \text{monus})$ and hence is a primitive recursive function. Thus $\chi_A = \mu [\min \circ (g \land h \circ (I_{1,2} \land \cdots \land I_{n,n+1} \land I_{n+1,n+2}))],^2$ is a recursive total function. Hence $A$ is a recursive $n$-ary relation.

Def: A language $L \subseteq T^*$ is said to be recursively enumerable (r.e.) if $e(L) = \emptyset$ or $e(L) = \{g(n) : n \in N \}$ for some recursive total function $g : N \rightarrow N$. Here $e(L)$ is a coding of the strings of $L$ into natural numbers.
It can be shown that \( c(L) \) is a recursively enumerable unary relation \( \iff c(L) \) is a semi-recursive relation. So we have the following correspondences:

- **MACHINES**
  1. Turing computable function
  2. Turing decidable relation
  3. Turing semi-decidable relation
  4. Turing decidable language
  5. Turing semi-decidable language

- **EXPRESSIONS**
  1. Recursive function
  2. Recursive relation
  3. Semi-recursive relation
  4. Recursive language
  5. Semi-recursive language
  6. Recursively enumerable language

A hierarchy of languages:

- \( A = \{a^n : n \geq 0\} \)
- \( B = \{a^n b^n : n \geq 0\} \)
- \( C = \{a^m b^m c^k : n, k \geq 1\} \)
- \( D = \{a^n b^n c^n : n \geq 1\} \)
- \( E \) is too complicated to describe easily.
- \( F = \{w \in \{a,b,c\}^* : M_u \text{ halts on } w\} \)
- \( G = F^c \)

Here \( M_u = \) a fixed universal Turing Machine.