Now that we have learned about composition and recursion, let us consider the functions that can be constructed with these operations. Let us define the following initial functions:

\[ s(x) = x + 1 \]
\[ n(x) = 0 \]
\[ u_i^n(x_1, \ldots, x_n) = x_i, \ 1 \leq i \leq n. \]

The functions \( u_i^n \) are called projection functions. For example, \( u_2^3(x_1, x_2, x_3) = x_2 \).

**Definition:** A class of total functions \( C \) is called a PRC (primitive recursively closed) class if
- the initial functions belong to \( C \),
- a function obtained from functions belonging to \( C \) by either composition or recursion also belongs to \( C \).

**Theorem 3.1:** The class of computable functions is a PRC class.

**Proof:** We already know through Theorems 1.1, 2.1, and 2.2 that applying composition or recursion to computable functions results in further computable functions. Therefore, we only have to show that the initial functions are computable. Obviously, \( s(x) = x + 1 \) is computed by
\[ Y \leftarrow X \\
Y \leftarrow Y + 1, \]
\( n(x) \) is computed by the empty program, and \( u_i^n(x_1, \ldots, x_n) \) is computed by
\[ Y \leftarrow X_i. \]

**Definition:** A function is called primitive recursive if it can be obtained from the initial functions by a finite number of applications of composition and recursion.

Obviously, it follows that:

**Corollary 3.2:** The class of primitive recursive functions is a PRC class.

Furthermore, we have:

**Theorem 3.3:** A function is primitive recursive if and only if it belongs to every PRC class.

**Proof:**

**Part I:** If a function belongs to every PRC class, then, by Corollary 3.2, it belongs to the class of primitive recursive functions.

**Part II:** Let \( f \) be a primitive recursive function and let \( C \) be some PRC class. We want to show that \( f \) belongs to \( C \).

Since \( f \) is a primitive recursive function, there is a list \( f_1, f_2, \ldots, f_n \) of functions such that \( f_n = f \) and each \( f_i \) in the list is either
- an initial function or
- can be obtained from preceding functions in the list by composition or recursion.

Obviously, the initial functions belong to the PRC class \( C \).
Applying composition or recursion to functions in \( C \) results in another function belonging to \( C \).
Thus each function in the list \( f_1, \ldots, f_n \) belongs to \( C \). Since \( f_n = f \), \( f \) belongs to \( C \).

**Corollary 3.4:** Every primitive recursive function is computable.

**Proof:** By the theorem just proved, every primitive recursive function belongs to the PRC class of computable functions.
Another Word on PRC Classes

A class of total functions $C$ is called a PRC class if
• the initial functions $n$, $s$, and $u$ belong to $C$ and
• a function obtained from functions belonging to $C$ by recursive or composition also belongs to $C$.

Notice that this definition does not demand all functions in $C$ to be obtained from $n$, $s$, and $u$ by recursion or composition.

There could be other functions in $C$, say $p$ and $q$, that cannot be obtained from $n$, $s$, and $u$. According to the definition, $C$ is then still a PRC class if all functions obtained from $n$, $s$, $u$, $p$, and $q$ by recursion or composition are also in $C$.

Now look at the definition of primitive recursive functions:

A function is called primitive recursive if it can be obtained from the initial functions by a finite number of applications of composition and recursion.

So here no additional functions such as $p$ and $q$ are allowed – all primitive recursive functions can be obtained from $n$, $s$, and $u$.

Therefore, the class of primitive recursive functions is the minimal PRC class. All primitive recursive functions are contained in every PRC class. However, PRC classes can contain additional functions (see previous slide).

Some Primitive Recursive Functions

So we have learned that every primitive recursive function is computable.

We will now look at some examples of primitive recursive functions.

In order to prove that a function is primitive recursive, we have to show how that function can be derived from the initial functions by using composition and recursion.

Remember? The recursive definition of a function looks like this:

$h(x_1, ..., x_n, 0) = f(x_1, ..., x_n)$

$h(x_1, ..., x_n, t + 1) = g(t, h(x_1, ..., x_n, t), x_1, ..., x_n)$.

If $f$ is a function of $k$ variables, $g_1, ..., g_k$ are functions of $n$ variables, and $h(x_1, ..., x_n) = f(g_1(x_1, ..., x_n), ..., g_k(x_1, ..., x_n))$.

Then $h$ is said to be obtained from $f$ and $g_1, ..., g_k$ by composition.

We defined the following initial functions:

$s(x) = x + 1$
$n(x) = 0$
$u^i(x_1, ..., x_n) = x_i, \ 1 \leq i \leq n.$