Composition

Example:
We know that $f(x_1, x_2) = x_1 + x_2$ is a computable function.
We also know that $g_1(x) = x^2$ and $g_2(x) = 3x$ are computable functions.
According to Theorem 1.1, the following function $h(x)$ must then also be computable:
$$h(x) = f(g_1(x), g_2(x)) = f(x^2, 3x) = x^2 + 3x$$

Recursion

Let $k$ be some fixed number and
$$h(0) = k$$
$$h(t + 1) = g(t, h(t)),$$
where $g$ is some given total function of two variables.
Then we say that $h$ is obtained from $g$ by primitive recursion, or simply recursion.

Theorem 2.1: Let $h$ be obtained as shown above, and let $g$ be computable. Then $h$ is also computable.

Proof:
Obviously, the function $f(x) = k$ is computable.
The program computing $f(x)$ simply consists of $k$ times the instruction $Y ← Y + 1$.
This gives us the macro $Y ← k$.
Now we can write a program that computes $h(x)$:
$$Y ← f(x_1, \ldots, x_n)$$
$$[A] \text{ IF } X = 0 \text{ GOTO E }$$
$$Y ← g(Z, Y)$$
$$Z ← Z + 1$$
$$X ← X - 1$$
$$\text{ GOTO A }$$

Recursion

Here is a similar, but more complicated type of recursion:
$$h(x_1, \ldots, x_n, 0) = f(x_1, \ldots, x_n)$$
$$h(x_1, \ldots, x_n, t + 1) = g(t, h(x_1, \ldots, x_n, t), x_1, \ldots, x_n).$$
Here we say that the function $h$ of $n + 1$ variables is obtained by primitive recursion (or recursion) from the functions $f$ (of $n$ variables) and $g$ (of $n + 2$ variables).

Theorem 2.2: Let $h$ be obtained from $f$ and $g$ as shown above, and let $f$ and $g$ be computable. Then $h$ is also computable.

Proof: The following program computes $h(x_1, \ldots, x_n, x_{n+1})$:
$$Y ← f(x_1, \ldots, x_n)$$
$$[A] \text{ IF } X_{n+1} = 0 \text{ GOTO E }$$
$$Y ← g(Z, Y, x_1, \ldots, x_n)$$
$$Z ← Z + 1$$
$$X_{n+1} ← X_{n+1} - 1$$
$$\text{ GOTO A }$$

PRC Classes

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:
$$s(x) = x + 1$$
$$n(x) = 0$$
$$u^i(x_1, \ldots, x_n) = x_i, \quad 1 ≤ i ≤ n.$$ The functions $u^i$ are called projection functions. For example, $u_2^3(x_1, x_2, x_3) = x_2$. 

September 26, 2017 Theory of Computation Lecture 7: Primitive Recursive Functions III
**PRC Classes**

**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 \(u^n(x_1, \ldots, x_n)\) is computed by
\[Y \leftarrow X_i\]

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

**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.