Composition

Let us combine computable functions in such a way that the output of one becomes an input to another. For example, we could combine the functions $f$ and $g$ to obtain a new function $h$:

$$h(x) = f(g(x))$$

Let us now take a more general view:

**Definition:** Let $f$ be a function of $k$ variables and let $g_1, \ldots, g_k$ be functions of $n$ variables. Let

$$h(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n)).$$

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

**Theorem 1.1:** If $h$ is obtained from the (partially) computable functions $f, g_1, \ldots, g_k$ by composition, then $h$ is (partially) computable.

**Proof:** The following program obviously computes $h$:

1. $Z_1 \leftarrow g_1(X_1, \ldots, X_n)$
2. $Z_k \leftarrow g_k(X_1, \ldots, X_n)$
3. $Y \leftarrow f(Z_1, \ldots, Z_k)$

If $f, g_1, \ldots, g_k$ are not only partially computable but are also total, then so is $h$. ■

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 $Y \leftarrow Y + 1$.

This gives us the macro $Y \leftarrow k$.

Now we can write a program that computes $h(x)$:

1. $Y \leftarrow k$
2. IF $X = 0$ GOTO E
3. $Y \leftarrow g(Z, Y)$
4. $Z \leftarrow Z + 1$
5. $X \leftarrow X - 1$
6. 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.
Recursion

Proof: The following program computes $h(x_1, \ldots, x_n, x_{n+1})$:

```
Y ← f(X_1, \ldots, X_n)
[A] IF X_{n+1} = 0 GOTO E
Y ← g(Z, Y, X_1, \ldots, X_n)
Z ← Z + 1
X_{n+1} ← X_{n+1} - 1
GOTO A ■
```

PRC Classes

Definition: A class of total functions $\mathcal{C}$ is called a PRC (primitive recursively closed) class if

- the initial functions belong to $\mathcal{C}$,
- a function obtained from functions belonging to $\mathcal{C}$ by either composition or recursion also belongs to $\mathcal{C}$.

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 ← X
Y ← Y + 1,
```

$n(x) = 0$ is computed by

```
Y ← Y + 1,
```

$u^n(x_1, \ldots, x_n) = x_i$, $1 \leq i \leq n$. The functions $u^n$ are called projection functions. For example, $u_2^3(x_1, x_2, x_3) = x_2$.

PRC Classes

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 $\mathcal{C}$ be some PRC class. We want to show that $f$ belongs to $\mathcal{C}$.

Since $f$ is a primitive recursive function, there is a list $f_1, f_2, \ldots, f_n$ of functions such that $f_i = f$ 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.
PRC Classes

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 $n, s, \text{ and } u$ belong to $C$ and
- a function obtained from functions belonging to $C$ by recursion or composition also belongs to $C$.

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

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

---

Another Word on PRC Classes

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 $n, s, \text{ and } u$.

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).