Primitive Rec, Ackerman's Function, Decidable, Undeciable, and Beyond
Exposition by William Gasarch

1 Primitive Recursive Functions

We would like to *formally* define some notion of "computable functions." We *attempt* to define a set of functions that contains only computable functions, and contains all of them. This attempt will fail, but the reasons for this are of interest. In a later section we will succeed.

In this attempt to define "computable functions" we try to use only basic operations (e.g. the operation "add 1"), and basic ways of putting functions together (e.g. composition). We do this to make the model as simple as possible, and to guarantee that all functions generated are computable.

Def 1.1 A function $f(x_1, \ldots, x_n)$ is *primitive recursive* if either:

1. f is the function that is always 0, i.e. $f(x_1, \ldots, x_n) = 0$; This is denoted by Z when the number of arguments is understood. This rule for deriving a primitive recursive function is called the Zero rule.

2. f is the successor function, i.e. $f(x_1, \ldots, x_n) = x_i + 1$; This rule for deriving a primitive recursive function is called the Successor rule.

3. f is the projection function, i.e. $f(x_1, \ldots, x_n) = x_i$; This is denoted by π_i when the number of arguments is understood. This rule for deriving a primitive recursive function is called the Projection rule.

4. f is defined by the composition of (previously defined) primitive recursive functions, i.e. if $g_1(x_1, \ldots, x_n)$, $g_2(x_1, \ldots, x_n)$, $\ldots$, $g_k(x_1, \ldots, x_n)$ are primitive recursive and $h(x_1, \ldots, x_k)$ is primitive recursive, then

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

is primitive recursive. This rule for deriving a primitive recursive function is called the Composition rule.

5. f is defined by recursion of two primitive recursive functions, i.e. if $g(x_1, \ldots, x_{n-1})$ and $h(x_1, \ldots, x_{n+1})$ are primitive recursive then the following function is also primitive recursive

$$\begin{aligned} f(x_1, \ldots, x_{n-1}, 0) &= g(x_1, \ldots, x_{n-1}) \\ f(x_1, \ldots, x_{n-1}, m+1) &= h(x_1, \ldots, x_{n-1}, m, f(x_1, \ldots, x_{n-1}, m)) \end{aligned}$$

   This rule for deriving a primitive recursive function is called the Recursion rule. It is a very powerful rule and is why these functions are called 'primitive recursive.'

To show some function is primitive recursive you build it up from these rules. Such a proof is called a derivation of that primitive recursive function.

We give some examples of primitive recursive functions. These examples will be given both rather formally (more formal than is really needed) and less formally. After these examples we shall always be less formal. The only reason to give the rather formal definition is to show that it can be done. In practice the informal one is just as informative (actually more so) and has the intuition that the formal one sorely lacks.

There are two points these examples are making: (1) LOTS of functions are primitive recursive. Most functions that you have dealt with in your lives have been primitive recursive. (2) It isn't that hard to show that functions are primitive recursive.

Example 1.2 $f(x) = x + 5$. Very formally we would say:

1. $S(x) = x + 1$ is primitive recursive by the successor rule.
2. $S(S(x))$ is primitive recursive by the composition rule, composing the function defined in 1 with the function defined in 1.
3. $S(S(S(x)))$ is primitive recursive by the composition rule, composing the function composed in 2 with the function defined in 1.
4. $S(S(S(S(x))))$ is primitive recursive by the composition rule, composing the function defined in 3 with the function defined in 1.
5. $S(S(S(S(S(x)))))$ is primitive recursive by the composition rule, composing the function defined in 4 with the function defined in 1.

Informally we would say: $f(x) = S(S(S(S(S(x)))))$ is primitive recursive by successor and several applications of composition.

Example 1.3 $f(x, y) = x + y$. Very formally we would say

1. $g(x) = \pi_1(x) = x$ is primitive recursive by the projection rule.
2. $\pi_3(x, y, z) = z$ is primitive recursive by the projection rule.
3. $S(x) = x + 1$ is primitive recursive by the successor rule.
4. $h(x, y, z) = S(\pi_3(x, y, z)) = z + 1$ is primitive recursive by the composition rule and composing S and π_3.
5. Define f using the recursion rule and g, h:

$$\begin{aligned} f(x, 0) &= g(x) = \pi_1(x) = x \\ f(x, n + 1) &= h(x, n, f(x, n)) = S(\pi_3(x, n, f(x, n))) = f(x, n) + 1 \end{aligned}$$

Informally, just say

$$\begin{aligned} f(x, 0) &= x \\ f(x, n + 1) &= f(x, n) + 1 \end{aligned}$$

The question arises, "how does one think of these definitions?" The key point is to think recursively. The thought process for defining plus might be "Well gee, $x + 0 = x$, that's easy enough, and $x + (y + 1) = (x + y) + 1$, so I can define $x + (y + 1)$ in terms of $x + y$." From that point it should be easy to see how to formalize that $f(x, y) = x + y$ is primitive recursive.

Now that we have plus is primitive recursive, we can use it to define other primitive recursive functions. Here we use it to define multiplication.

Example 1.4 $f(x, y) = xy$ is primitive recursive via

$$\begin{aligned} f(x, 0) &= 0 \\ f(x, y + 1) &= f(x, y) + x \end{aligned}$$