rec.1  Examples of Primitive Recursive Functions

We already have some examples of primitive recursive functions: the addition and multiplication functions add and mult. The identity function $\text{id}(x) = x$ is primitive recursive, since it is just $P^1_0$. The constant functions $\text{const}_n(x) = n$ are primitive recursive since they can be defined from zero and $\text{succ}$ by successive composition. This is useful when we want to use constants in primitive recursive definitions, e.g., if we want to define the function $f(x) = 2 \cdot x$ we can obtain it by composition from $\text{const}_n(x)$ and multiplication as $f(x) = \text{mult}(\text{const}_2(x), P^1_0(x))$. We'll make use of this trick from now on.

Proposition rec.1. The exponentiation function $\exp(x, y) = x^y$ is primitive recursive.

Proof. We can define $\exp$ primitive recursively as

\[
\begin{align*}
\exp(x, 0) &= 1 \\
\exp(x, y + 1) &= \text{mult}(x, \exp(x, y)).
\end{align*}
\]

Strictly speaking, this is not a recursive definition from primitive recursive functions. Officially, though, we have:

\[
\begin{align*}
\exp(x, 0) &= f(x) \\
\exp(x, y + 1) &= g(x, y, \exp(x, y)).
\end{align*}
\]

where

\[
\begin{align*}
f(x) &= \text{succ}(\text{zero}(x)) = 1 \\
g(x, y, z) &= \text{mult}(P^3_0(x, y, z), P^3_2(x, y, z)) = x \cdot z
\end{align*}
\]

and so $f$ and $g$ are defined from primitive recursive functions by composition. □

Proposition rec.2. The predecessor function $\text{pred}(y)$ defined by

\[
\text{pred}(y) = \begin{cases} 
0 & \text{if } y = 0 \\
y - 1 & \text{otherwise}
\end{cases}
\]

is primitive recursive.

Proof. Note that

\[
\begin{align*}
\text{pred}(0) &= 0 \\
\text{pred}(y + 1) &= y.
\end{align*}
\]

This is almost a primitive recursive definition. It does not, strictly speaking, fit into the pattern of definition by primitive recursion, since that pattern requires
at least one extra argument $x$. It is also odd in that it does not actually use \textit{pred}(y) in the definition of \textit{pred}(y + 1). But we can first define \textit{pred}'(x, y) by

\[
\text{pred}'(x, 0) = \text{zero}(x) = 0, \\
\text{pred}'(x, y + 1) = P^1_1(x, y, \text{pred}'(x, y)) = y. 
\]

and then define \textit{pred} from it by composition, e.g., as \textit{pred}(x) = \text{pred}'(\text{zero}(x), P^1_0(x))$.\hfill\Box

\textbf{Proposition rec.3.} \textit{The factorial function} $\text{fac}(x) = x! = 1 \cdot 2 \cdot 3 \cdots x$ \textit{is primitive recursive}.

\textit{Proof.} The obvious primitive recursive definition is

\[
\text{fac}(0) = 1 \\
\text{fac}(y + 1) = \text{fac}(y) \cdot (y + 1).
\]

Officially, we have to first define a two-place function $h$

\[
h(x, 0) = \text{const}_1(x) \\
h(x, y) = g(x, y, h(x, y))
\]

where $g(x, y, z) = \text{mult}(P^2_2(x, y, z), \text{succ}(P^1_1(x, y, z)))$ and then let

\[
\text{fac}(y) = h(P^1_0(y), P^1_0(y))
\]

From now on we’ll be a bit more laissez-faire and not give the official definitions by composition and primitive recursion. \hfill\Box

\textbf{Proposition rec.4.} \textit{Truncated subtraction}, $x \dot{-} y$, \textit{defined by}

\[
x \dot{-} y = \begin{cases} 
0 & \text{if } x > y \\
x - y & \text{otherwise}
\end{cases}
\]

is primitive recursive.

\textit{Proof.} We have:

\[
x \dot{-} 0 = x \\
x \dot{-} (y + 1) = \text{pred}(x \dot{-} y)
\]

\textbf{Proposition rec.5.} \textit{The distance between $x$ and $y$, $|x - y|$, is primitive recursive}.

\textit{Proof.} We have $|x - y| = (x \dot{-} y) + (y \dot{-} x)$, so the distance can be defined by composition from $+$ and $\dot{-}$, which are primitive recursive. \hfill\Box
Proposition rec.6. The maximum of \( x \) and \( y \), \( \text{max}(x, y) \), is primitive recursive.

Proof. We can define \( \text{max}(x, y) \) by composition from \(+\) and \( \dot{-} \) by

\[
\text{max}(x, y) = x + (y \dot{-} x).
\]

If \( x \) is the maximum, i.e., \( x \geq y \), then \( y \dot{-} x = 0 \), so \( x + (y \dot{-} x) = x + 0 = x \). If \( y \) is the maximum, then \( y \dot{-} x = y - x \), and so \( x + (y \dot{-} x) = x + (y - x) = y \).

Proposition rec.7. The minimum of \( x \) and \( y \), \( \text{min}(x, y) \), is primitive recursive.

Proof. Exercise.

Problem rec.1. Prove Proposition rec.7.

Problem rec.2. Show that

\[
f(x, y) = 2^{(2^{x})^y}2^y
\]

is primitive recursive.

Problem rec.3. Show that integer division \( d(x, y) = \lfloor x/y \rfloor \) (i.e., division, where you disregard everything after the decimal point) is primitive recursive. When \( y = 0 \), we stipulate \( d(x, y) = 0 \). Give an explicit definition of \( d \) using primitive recursion and composition.

Proposition rec.8. The set of primitive recursive functions is closed under the following two operations:

1. Finite sums: if \( f(\vec{x}, z) \) is primitive recursive, then so is the function

\[
g(\vec{x}, y) = \sum_{z=0}^{y} f(\vec{x}, z).
\]

2. Finite products: if \( f(\vec{x}, z) \) is primitive recursive, then so is the function

\[
h(\vec{x}, y) = \prod_{z=0}^{y} f(\vec{x}, z).
\]

Proof. For example, finite sums are defined recursively by the equations

\[
g(\vec{x}, 0) = f(\vec{x}, 0)
g(\vec{x}, y + 1) = g(\vec{x}, y) + f(\vec{x}, y + 1).
\]

\[\square\]