Primitive Recursive Functions

Robert Rynasiewicz
Mathematical Logic II

Spring 2022

Some Special Functions

Let's forget about formal systems for awhile. Focus on just these three (types) of functions.

- the 0-ary constant function 0
- the *successor* function S
- the *projection* functions: for each $n \in \mathbb{N}$ and each k, $1 \leq k \leq n$, the function $I^n_k : \mathbb{N}^n \to \mathbb{N}$ is defined

$$I^n_k(x_1, \ldots, x_k, \ldots, x_n) = x_k.$$

All primitive recursive functions are built out of these three.

Q. But what are the techniques of building?

Deviation from Peter Smith?

Smith says the recursive functions are built from S, the projection functions and the *unary* zero function $Z(n) = 0$ for all $n \in \mathbb{N}$.

We'll see why we need the 0-ary constant function 0 later on.

For now just note that if we have the 0-ary constant function 0, we can get the unary zero function Z by recursion:

$$\begin{aligned} Z(0) &= 0 \\ Z(S(y)) &= I_2^2(y, Z(y)) \end{aligned}$$

We'll see why we put it in exactly this form later on.

Composition of Functions

Composition of Functions

- The standard definition for functions of one variable: Given $g : A \to B$ and $h : B \to C$, then $h \circ g : A \to C$ is defined $(h \circ g)(x) = h(g(x))$ for all $x \in A$.
- We want to consider functions of many variables.
- Simplifying matters is that the domain will always be a Cartesian product of the natural numbers.
- To condense notation we let $\vec{x}$ stand for an arbitrary tuple. (The length of $\vec{x}$ will be determined from context.)

Composition of Functions (cont.)

Defn. Let h be a k-ary function and $g_1 \dots g_k$ be n-ary functions. Then

$$f(\vec{x}) = h(g_1(\vec{x}), \dots, g_k(\vec{x}))$$

is the n-ary function obtained from h and $g_1, \dots, g_k$ by composition.

Examples:

- $f(x, y) = S(y)$ is constructed explicitly as follows:

$$f(x, y) = S(I_2^2(x, y)).$$

- $f(x, y, z) = (x \times y) + (y \times z)$ is built up by composition (in two steps) as follows, using prefix notation:

$$f(x, y, z) = +(\times(I_1^3(x, y, z), I_2^3(x, y, z)), \times(I_2^3(x, y, z), I_3^3(x, y, z))).$$

Composition of Functions (cont.)

To go through the last example in complete detail:

Let

$$h_1(x, y, z) = \times(I_1^3(x, y, z), I_2^3(x, y, z))$$

and

$$h_2(x, y, z) = \times(I_2^3(x, y, z), I_3^3(x, y, z)).$$

Then

$$f(x, y, z) = +(h_1(x, y, z), h_2(x, y, z)).$$

Primitive Recursion (one variable)

Recursion for functions of a single variable. Simplest case is when we are dealing with a single variable., e.g., defining factorial $f(n)$ by recursion:

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

If we set $h(y, z) = z \cdot S(y)$, then this has the general form:

$$\begin{aligned} f(0) &= \text{0-ary function (in this case S(0))} \\ f(S(y)) &= h(y, f(y)) \end{aligned}$$

Primitive Recursion (several variables)

Recursion for functions of several variables. To generalize, we just consider that f and h may be functions of additional variables $\vec{x} = \langle x_1, \ldots, x_n \rangle$ and that $f(\vec{x}, 0)$ may be some function g of $\vec{x}$.
Defn. f is defined from g and h *by primitive recursion* if

$$\begin{aligned} f(\vec{x}, 0) &= g(\vec{x}) \\ f(\vec{x}, S(y)) &= h(\vec{x}, y, f(\vec{x}, y)). \blacksquare \end{aligned}$$

This is the form that we've seen for addition, setting $\vec{x} = x$:

$$\begin{aligned} +(x, 0) &= I_1^1(x) \\ +(x, S(y)) &= h(x, y, +(x, y)) \end{aligned}$$

where $h(x, y, z) = S(I_3^3(x, y, z))$.

Definition of the Primitive Recursive Functions

The *primitive recursive functions* are defined inductively:

- The 0-ary constant function 0 is primitive recursive.
- The successor function S is primitive recursive.
- The projection functions I_k^n are primitive recursive.
- The composition of recursive functions is recursive, i.e., if $h : \mathbb{N}^k \to \mathbb{N}$ is primitive recursive and $g_1(\vec{x}), \ldots, g_k(\vec{x})$ are primitive recursive, then $h(g_1(\vec{x}), \ldots, g_k(\vec{x}))$ is primitive recursive.
- If g and h are primitive recursive and f is defined from g and h by primitive recursion, then f is primitive recursive.

Survey of Some Primitive Recursive Functions

- The unary zero function $Z(n)$ is primitive recursive. (Go back a number of slides: $Z(0) = 0$, $Z(S(n)) = I_2^2(n, Z(n))$.)
- Addition is primitive recursive since $S(I_3^3(x, y, z))$ is the composition of recursive functions and addition is defined from $I_1^2(x, y)$ and $S(I_3^3(x, y, z))$ by primitive recursion. Explicitly,

$$\begin{aligned} +(x, 0) &= I_1^2(x, 0) \\ +(x, S(y)) &= S(I_3^3(x, y, +(x, y))). \end{aligned}$$

Survey of Some Primitive Recursive Functions (cont.)

- Multiplication is primitive recursive since it is defined from 0 and h by primitive recursion as follows:

$$\begin{aligned} \times(x,0) &= Z(x) \\ \times(x,Sy) &= h(x,y,\times(x,y)), \end{aligned}$$

where $h(x,y,z) = +(I_3^3(x,y,z), I_1^3(x,y,z))$.

Thus,

$$\begin{aligned} x \cdot 0 &= x \\ x \cdot S(y) &= x \cdot y + x. \end{aligned}$$

Survey of Some Primitive Recursive Functions (cont.)

- Factorial is defined from the 0-ary function $S(0)$ (a.k.a. 1) and $h(y,z)$ by

$$\begin{aligned} f(0) &= S(0) \\ f(S(y)) &= h(y, f(y)), \end{aligned}$$

  where $h(y,z) = \times(S(I_1^2(y,z)), I_2^2(y,z))$.
- Exponentiation:

$$\begin{aligned} exp(x,0) &= S(Z(x)) \\ exp(x,S(y)) &= h(x,y,exp(x,y)), \end{aligned}$$

  where $h(x,y,z) = \times(I_1^3(x,y,z), I_3^3(x,y,z))$.

Survey of Some Primitive Recursive Functions (cont.)

- Truncated Predecessor:

$$\begin{aligned} Pred(0) &= 0 \\ Pred(S(y)) &= I_1^2(y, Pred(y)) \end{aligned}$$

- Truncated Subtraction

$$\begin{aligned} x -' 0 &= I_1^1(x) \\ x -' S(y) &= Pred(I_3^3(x, y, x -' y)) \end{aligned}$$

- Absolute Difference

$$| x - y | = (x -' y) + (y -' x)$$

Computability of Primitive Recursive Functions

Primitive recursive functions are pretty obviously computable. You might say, they're computable if anything is. But it is of some interest to see how they fit into our intuitive characterization of computability in terms of a finite list of executable instructions, i.e., a computer program in a language that has only 0, S, and the projection functions built in.

- Composition is handled by feeding the outputs of the g functions into the h function.

Computability of Primitive Recursive Functions (cont.)

• Primitive recursion is handled by *for* loops as follows. Take the function to be $f(x,0) = g(x); f(x,S(y)) = h(x,y,f(x,y))$ where it is assumed that there are programs for computing $g(x)$ and $h(x,y,z)$.

Let *func* be a memory register. Then the following program computes $f(m,n)$.

1. $func := g(m)$
2. for $y = 1$ to n do
3. $\quad\quad func := h(m,y,func)$
4. end do

Computability of Primitive Recursive Functions (cont.)

Claim* Any primitive recursive function is computable by (perhaps nested) *for* loops. Conversely, any function computable by nested for loops is primitive recursive.

This leads one to expect that there are computable functions that are not primitive recursive, since programming languages have **do while** and **do until** loops that don't involve a fixed upped bound on the number of loops. This we can show. But first:

Claim*. The set of primitive recursive functions is effectively enumerable.

Proof 1. Effectively enumerate the set of programs that use only **for** loops. ■

The Primitive Recursive Functions Are E.E. (cont.)

Proof 2.

- Effectively enumerate 0, S, and the projection functions.
- Successively run through the list formulating all p.r. functions that can be constructed by a single application of composition or p.r. to the functions so far encountered. At each stage this yields a finite number of new p.r. functions.
- The functions within each stage of formation can be well-ordered, first by increasing arity, and then within a given arity n by consulting the dictionary order on the functions consider as $(n+1)$-ary relations. ■

Some Computable Functions Are Not P.R.

Theorem*. There exists an effectively computable function that is not primitive recursive.

Proof. Let $f_0, f_1, f_2, \ldots$ be an effective enumeration of the p.r. functions. Define $g(n) = f_n(n) + 1$. Obviously, g is effectively computable. But g is not p.r., since if it were, $g = f_k$ for some $k \in \mathbb{N}$. But $g(k) = f_k(k) + 1 \neq f_k(k)$. ■

	0	1	2	$\cdots$
f_0	$\underline{f_0(0)}$	$f_0(1)$	$f_0(2)$	$\cdots$
f_1	$f_1(0)$	$\underline{f_1(1)}$	$f_1(2)$	$\cdots$
f_2	$f_2(0)$	$f_2(1)$	$\underline{f_2(2)}$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\searrow$

Primitive Recursive Properties and Relations

Defn. A relation is primitive recursive iff its characteristic function is primitive recursive.

Thus, a property is p.r. iff its characteristic function is p.r., since a property is a unary relation.

Example. $\{0\}$ is p.r. It's characteristic function is

$$\chi_{\{0\}}(n) = \begin{cases} 1 & \text{if n=0} \\ 0 & \text{otherwise} \end{cases}$$

We can define $\chi_{\{0\}}$ by primitive recursion as follows.

$$\begin{aligned} \chi_{\{0\}}(0) &= 1 \\ \chi_{\{0\}}(S(y)) &= Z(y). \end{aligned}$$

Primitive Recursive Properties and Relations (cont.)

Example. The set E of even numbers is p.r. We define its characteristic function χ_E by primitive recursion as follows.

$$\begin{aligned} \chi_E(0) &= 1 \\ \chi_E(S(y)) &= \chi_{\{0\}}(\chi_E(y)). \end{aligned}$$

Example. $\{\langle m, n\rangle \mid m = n\}$ is p.r. It's characteristic function $\chi_=$ is defined by:

$$\chi_=(x, y) = \chi_{\{0\}}(\mid x - y \mid).$$

Bounded Minimization

Notation. The expression μx is to be read as "the least x s.t.".

Def: Bounded Minimization. Let $P(x)$ be some condition on x. Then the function $f(n) = (\mu x \leq n)P(x)$ takes the values

$$f(n) =_{df} \begin{cases} \text{the least } x \leq n \text{ s.t. } P(x) & \text{if } \exists x \leq n \text{ s.t. } P(x) \\ n & \text{otherwise.} \end{cases}$$

Generalizing, the function $f(n) = (\mu x \leq g(n))P(x)$ takes the values

$$f(n)_{df} = \begin{cases} \text{the least } x \leq g(n) \text{ s.t. } P(x) & \text{if } \exists x \leq g(n) \text{ s.t. } P(x) \\ g(n) & \text{otherwise.} \end{cases}$$

Bounded Minimization (cont.)

Example. Let $P(x)$ be 'x is prime'. Then $f(n) = (\mu x \leq n)P(x)$ takes on the values:

$$f(0) = 0,\ f(1) = 1,\ f(n) = 2 \text{ if } n \geq 2.$$

Example. Let $P(x)$ be as before and $g(n) = n \dot{-} 5$. Then we have

$$f(n) = \begin{cases} 0 & \text{if } n \leq 5 \\ 1 & \text{if } n = 6 \\ 2 & \text{otherwise} \end{cases}$$

Definition by Cases

Defn. Let $g_0, \ldots, g_k$ be p.r. functions and $C_0(x), \ldots, C_k(x)$ mutually exclusive p.r. properties, and let m be a fixed number. Then f is said to be *defined by cases from other p.r. functions* if

$$f(n) = \begin{cases} g_0(n) & \text{if } C_0(n) \\ \vdots & \vdots \\ g_k(n) & \text{if } C_k(n) \\ m & \text{otherwise,} \end{cases}$$

where m is some preselected natural number.

Some Notation

Let sg be the characteristic function of the set of non-zero natural numbers, i.e.,

$$sg(n) = \begin{cases} 0 & \text{n} = 0 \\ 1 & \text{otherwise} \end{cases}$$

The function sg is p.r. as can be seen from the following formulation:

$$\begin{aligned} sg(0) &= 0 \\ sg(S(y)) &= S(Z(sg(y))) \end{aligned}$$

Finally, let $\overline{sg} = \chi_{\{0\}}$, i.e., the characteristic function of the singleton set consisting of 0, which we showed to be p.r. earlier.
We now give a sequence of lemmas for constructing further p.r. functions and relations.

Lemma A

Lemma A. If $f(x_1, \ldots, x_n)$ is a p.r. function, then $\{\langle x_1, \ldots, x_n, f(x_1, \ldots, x_n)\rangle\}$ is a p.r. relation, i.e., its characteristic function is p.r.

Proof. Let χ_f be the characteristic function of the relation, i.e.,

$$\chi_f(x_1, \ldots, x_n, y) = \begin{cases} 1 & \text{if } y = f(x_1, \ldots, x_n) \\ 0 & \text{otherwise.} \end{cases}$$

χ_f is p.r., since

$$\chi_f(x_1, \ldots, x_n, y) = \overline{sg}(\mid y - f(x_1, \ldots, x_n) \mid),$$

and is thus the composition of primitive recursive functions.

Lemma B

Lemma B. Boolean (i.e., truth-functional) combinations of p.r. properties or relations are p.r.

Proof. Suppose that $P(x)$ is p.r. with characteristic function χ_P. Then the characteristic function $\chi_{\neg P}$ of $\neg P(x)$ is

$$\chi_{\neg P}(n) = \overline{sg}(\chi_P(n)).$$

Suppose both $P(x)$ and $Q(x)$ are p.r. Then the characteristic function $\chi_{(P \wedge Q)}$ of the conjunction $P(x) \wedge Q(x)$ is just

$$\chi_{(P \wedge Q)}(n) = \chi_P(n) \cdot \chi_Q(n).$$

All other Boolean combinations can be gotten by combinations of these two. ■

Lemma C

Lemma C. Any property or relation defined by bounded quantification from a p.r. property or relation is p.r.

Proof. Suppose that $K(n)$ is given by definition as $(\forall x \leq n)P(x)$, where $P(x)$ is p.r. Then the characteristic function χ_K of $K(n)$ is given by

$$\chi_K(n) = \prod_{i=0}^{n} \chi_P(i).$$

We need only assure ourselves that this is p.r. The generalized product can be defined by primitive recursion as follows.

$$\begin{aligned} \chi_K(0) &= \chi_P(0) \\ \chi_K(S(n)) &= \chi_K(n) \cdot \chi_P(S(n)). \end{aligned}$$

Lemma C (cont.)

If $K'(n)$ is defined by $(\forall x \leq f(n))P(x)$, where f as well is p.r., then

$$\chi_{K'}(n) = \prod_{i=0}^{f(n)} \chi_P(i),$$

and

$$\chi_{K'}(n) = \chi_K(f(n)),$$

which is the composition of p.r. functions.

The case of bounded existential quantification is left as an EXERCISE. ■

Lemma D

Lemma D. Suppose that $P(x)$ is a p.r. property and g(n) a p.r. function. Then (a) the function $f(n) = (\mu x \leq n)P(x)$ is p.r., and (b) the function $f'(n) = (\mu x \leq g(n))P(x)$ is p.r.

Proof. (a) Let the property $J(n)$ be defined by $(\exists x \leq n)P(x)$. Consider the characteristic function $\chi_{\neg J}$ of its negation. The function f can be rewritten as

$$\begin{aligned} f(0) &= 0 \\ f(n+1) &= \chi_{\neg J}(0) + \chi_{\neg J}(1) + \cdots + \chi_{\neg J}(n-1) + \chi_{\neg J}(n) \end{aligned}$$

For $\chi_{\neg J}(i)$ takes on the value 1 until we reach a k s.t. $P(k)$, where it takes on the value 0 and stays 0 thereafter. Thus we need only write this in a form that is explicitly primitive recursion, viz.,

Lemma D (cont.)

$$\begin{aligned} f(0) &= 0 \\ f(S(n)) &= f(n) + \chi_{\neg J}(n). \end{aligned}$$

For part (b), let $J'(n)$ be defined by $(\exists x \leq g(n))P(x)$. Again, as i increases, $\chi_{\neg J'}$ takes on the value 1 until we reach a k s.t. $P(k)$, where it takes on the value 0 and stays zero thereafter, only now i can increase up through $g(n)$. So, $f'(n) = f(g(n))$. ■

Lemma E

Lemma E. A function defined by cases from p.r. functions is p.r.

Pf. Suppose f is defined

$$f(n) =_{df} \begin{cases} g_0(n) & \text{if } C_0(n) \\ \vdots & \vdots \\ g_k(n) & \text{if } C_k(n) \\ m & \text{otherwise,} \end{cases}$$

where $g_1, \ldots, g_k$ and $C_0, \ldots, C_k$ are all p.r. (and the C_i's are mutually exclusive). Let $P(n)$ be the property $\neg C_0(n) \wedge \cdots \wedge \neg C_k(n)$. Then

$$f(n) = g_0(n) \cdot \chi_{C_0}(n) + \cdots g_k(n) \cdot \chi_{C_k}(n) + m \cdot \chi_P(n). \quad \blacksquare$$

More Properties, Relations, and Functions That Are P.R.

Claim. $=$ and $\leq$ are both p.r.

Pf. We saw earlier that

$$\chi_{=}(x, y) = \overline{sg}(\mid x - y \mid).$$

For $\leq$ we have

$$\chi_{\leq}(x, y) = \overline{sg}(x -' y).\blacksquare$$

Claim. m divides n, i.e, $m \mid n$, is p.r.

Pf. $m \mid n$ iff $(\exists x \leq n)(m \neq 0 \wedge x \cdot m = n)$. Multiplication is p.r., and hence so is the relation $\{\langle x, m, n\rangle \mid x \cdot m = n\}$. $\blacksquare$

Primes and P.R. Properties

Claim. *Prime*(n) is p.r.

Pf. *Prime*(n) is equivalent to

$$2 \leq n \wedge (\forall x \leq n)(x \mid n \rightarrow x = 1 \vee x = n).$$

This property has been constructed from p.r. properties using only truth-functional connectives and bounded quantification. ■

Primes and P.R. Properties (cont.)

Defn. Let $\pi_0, \pi_1, \ldots, \pi_n$ be the enumeration of the primes in increasing value. Then $\pi(n) =_{df} \pi_n$ is the $(n+1)$th prime.

Claim. For each $k \in \mathbb{N}$, $\pi(k+1) \leq \pi(k)! + 1$.

Proof. Obviously, $\pi(k) < \pi(k)! + 1$. So, if $\pi(k)! + 1$ is prime, then $\pi(k)! + 1 = \pi(m)$ for some $m > k$ and hence $\pi(k+1) \leq \pi(k)! + 1$. On the other hand, if $\pi(k)! + 1$ is not prime, then $p \mid \pi(k)! + 1$ for some prime $p < \pi(k)! + 1$. But $p > \pi(k)$, since any prime $q \leq \pi(k)$ divides $\pi(k)!$ evenly but not $\pi(k)! + 1$. ■

Primes and P.R. Properties (cont.)

Claim. $\pi(n)$ is p.r.

Proof. Since $\pi(k+1) \leq \pi(k)! + 1$ for each k, let

$$h(m) = (\mu x \leq m! + 1)(m < x \wedge Prime(x)).$$

The function $h(m)$ just gives us the least prime greater than m (where m may be, but need not be itself prime). Then we can define $\pi(n)$ by primitive recursion:

$$\begin{aligned} \pi(0) &= 2 \\ \pi(S(n)) &= h(\pi(n)). \ \blacksquare \end{aligned}$$

Primes and P.R. Properties (cont.)

Claim. Let $exf(n, i)$ be the (possibly zero) exponent of $\pi(i)$ in the prime factorization of n. Then $exf(n, i)$ is p.r.

Proof. Since no exponent in the prime factorization of n is greater than n,

$$exf(n, i) = (\mu x \leq n)((\pi(i)^x \mid n) \wedge (\pi(i)^{x+1} \nmid n)).$$

Primes and P.R. Properties (cont.)

Defn. Let

$$len(n) = \begin{cases} 0 & \text{if } n < 2 \\ \text{the number of prime factors of n} & \text{otherwise.} \end{cases}$$

Claim. $len(n)$ is p.r.

Pf. We build up $len(n)$ incrementally.

- Define $pf(m, n)$ to be $(Prime(m) \wedge (m \mid n))$.
- Let $p(m, n) = \chi_{pf}(m, n)$.

Primes and P.R. Properties (cont.)

- Note that $\mathit{len}(n) = p(0, n) + p(1, n) + \cdots + p(n, n)$.
- To capture this by primitive recursion first let

$$\begin{aligned} \ell(n, 0) &= p(0, n) \\ \ell(n, S(y)) &= \ell(n, y) + p(S(y), n). \end{aligned}$$

- Then set $\mathit{len}(n) = \ell(n, n)$. ■

Scholium. $\mathit{len}(n)$ stands for 'length of n', which you can think of as referring directly to the length of the prime factorization of n. But later on, when the Gödel numbers of wffs will be formed by multiplying primes exponentiated to the Gödel numbers of individual symbols, $\mathit{len}(n)$ applied to the Gödel number n of a wff yields for us the length of the wff.