Partial Recursive Functions

Aim

A more abstract, machine-independent description of the collection of computable partial functions than provided by register/Turing machines:

> they form the smallest collection of partial functions containing some basic functions and closed under some fundamental operations for forming new functions from old—composition, primitive recursion and minimization.

The characterization is due to Kleene (1936), building on work of Gödel and Herbrand.

Examples of recursive definitions

$$\begin{cases} f_1(0) & \equiv 0 \\ f_1(x+1) & \equiv f_1(x) + (x+1) \end{cases}$$

$f_1(x) =$ sum of $0, 1, 2, \ldots, x$

Examples of recursive definitions

$$\begin{cases} f_1(0) & \equiv 0 \\ f_1(x+1) & \equiv f_1(x) + (x+1) \end{cases}$$

$f_1(x) =$ sum of $0, 1, 2, \ldots, x$

$$\begin{cases} f_2(0) & \equiv 0 \\ f_2(1) & \equiv 1 \\ f_2(x+2) & \equiv f_2(x) + f_2(x+1) \end{cases}$$

$f_2(x) = x$th Fibonacci number

Examples of recursive definitions

$$\begin{cases} f_1(0) & \equiv 0 \\ f_1(x+1) & \equiv f_1(x) + (x+1) \end{cases}$$

$f_1(x) =$ sum of $0, 1, 2, \ldots, x$

$$\begin{cases} f_2(0) & \equiv 0 \\ f_2(1) & \equiv 1 \\ f_2(x+2) & \equiv f_2(x) + f_2(x+1) \end{cases}$$

$f_2(x) = x$th Fibonacci number

$$\begin{cases} f_3(0) & \equiv 0 \\ f_3(x+1) & \equiv f_3(x+2) + 1 \end{cases}$$

$f_3(x)$ undefined except when $x = 0$

Examples of recursive definitions

$$\begin{cases} f_1(0) & \equiv 0 \\ f_1(x+1) & \equiv f_1(x) + (x+1) \end{cases}$$

$f_1(x) =$ sum of $0, 1, 2, \ldots, x$

$$\begin{cases} f_2(0) & \equiv 0 \\ f_2(1) & \equiv 1 \\ f_2(x+2) & \equiv f_2(x) + f_2(x+1) \end{cases}$$

$f_2(x) = x$th Fibonacci number

$$\begin{cases} f_3(0) & \equiv 0 \\ f_3(x+1) & \equiv f_3(x+2) + 1 \end{cases}$$

$f_3(x)$ undefined except when $x = 0$

$$f_4(x) \equiv \textit{if } x > 100 \textit{ then } x - 10 \textit{ else } f_4(f_4(x+11))$$

f_4 is McCarthy's "91 function", which maps x to 91 if $x \leq 100$ and to $x - 10$ otherwise

Primitive recursion

Theorem. Given $f \in \mathbb{N}^n \rightharpoonup \mathbb{N}$ and $g \in \mathbb{N}^{n+2} \rightharpoonup \mathbb{N}$, there is a unique $h \in \mathbb{N}^{n+1} \rightharpoonup \mathbb{N}$ satisfying

$$\begin{cases} h(\vec{x}, 0) & \equiv f(\vec{x}) \\ h(\vec{x}, x+1) & \equiv g(\vec{x}, x, h(\vec{x}, x)) \end{cases}$$

for all $\vec{x} \in \mathbb{N}^n$ and $x \in \mathbb{N}$.

We write $\rho^n(f, g)$ for h and call it the partial function defined by primitive recursion from f and g.

Primitive recursion

Theorem. Given $f \in \mathbb{N}^n \rightharpoonup \mathbb{N}$ and $g \in \mathbb{N}^{n+2} \rightharpoonup \mathbb{N}$, there is a unique $h \in \mathbb{N}^{n+1} \rightharpoonup \mathbb{N}$ satisfying

$$(*) \begin{cases} h(\vec{x}, 0) & \equiv f(\vec{x}) \\ h(\vec{x}, x+1) & \equiv g(\vec{x}, x, h(\vec{x}, x)) \end{cases}$$

for all $\vec{x} \in \mathbb{N}^n$ and $x \in \mathbb{N}$.

Proof (sketch). *Existence*: the set

$$h \triangleq \{(\vec{x}, x, y) \in \mathbb{N}^{n+2} \mid \exists y_0, y_1, \ldots, y_x \\ f(\vec{x}) = y_0 \wedge (\textstyle\bigwedge_{i=0}^{x-1} g(\vec{x}, i, y_i) = y_{i+1}) \wedge y_x = y\}$$

defines a partial function satisfying $(*)$.

Uniqueness: if h and h' both satisfy $(*)$, then one can prove by induction on x that $\forall \vec{x}\ (h(\vec{x}, x) = h'(\vec{x}, x))$.

Example: addition

Addition $add \in \mathbb{N}^2 \to \mathbb{N}$ satisfies:

$$\begin{cases} add(x_1, 0) & \equiv x_1 \\ add(x_1, x+1) & \equiv add(x_1, x) + 1 \end{cases}$$

So $add = \rho^1(f, g)$ where $\begin{cases} f(x_1) & \triangleq x_1 \\ g(x_1, x_2, x_3) & \triangleq x_3 + 1 \end{cases}$

Note that $f = \mathtt{proj}_1^1$ and $g = \mathtt{succ} \circ \mathtt{proj}_3^3$; so add can be built up from basic functions using composition and primitive recursion: $add = \rho^1(\mathtt{proj}_1^1, \mathtt{succ} \circ \mathtt{proj}_3^3)$.

Example: predecessor

Predecessor $\mathit{pred} \in \mathbb{N} \to \mathbb{N}$ satisfies:

$$\begin{cases} \mathit{pred}(0) & \equiv 0 \\ \mathit{pred}(x+1) & \equiv x \end{cases}$$

So $\mathit{pred} = \rho^0(f, g)$ where $\begin{cases} f() & \triangleq 0 \\ g(x_1, x_2) & \triangleq x_1 \end{cases}$

Thus pred can be built up from basic functions using primitive recursion: $\mathit{pred} = \rho^0(\mathtt{zero}^0, \mathtt{proj}^2_1)$.

Example: multiplication

Multiplication $\mathit{mult} \in \mathbb{N}^2 \to \mathbb{N}$ satisfies:

$$\begin{cases} \mathit{mult}(x_1, 0) & \equiv 0 \\ \mathit{mult}(x_1, x+1) & \equiv \mathit{mult}(x_1, x) + x_1 \end{cases}$$

and thus $\mathit{mult} = \rho^1(\mathtt{zero}^1, \mathit{add} \circ (\mathtt{proj}^3_3, \mathtt{proj}^3_1))$.

So mult can be built up from basic functions using composition and primitive recursion (since add can be).

Definition. A [partial] function f is primitive recursive ($f \in \mathbf{PRIM}$) if it can be built up in finitely many steps from the basic functions by use of the operations of composition and primitive recursion.

In other words, the set $\mathbf{PRIM}$ of primitive recursive functions is the smallest set (with respect to subset inclusion) of partial functions containing the basic functions and closed under the operations of composition and primitive recursion.

Definition. A [partial] function f is primitive recursive ($f \in \mathbf{PRIM}$) if it can be built up in finitely many steps from the basic functions by use of the operations of composition and primitive recursion.

Every $f \in \mathbf{PRIM}$ is a total function, because:

- all the basic functions are total
- if $f, g_1, \ldots, g_n$ are total, then so is $f \circ (g_1, \ldots, g_n)$ [why?]
- if f and g are total, then so is $\rho^n(f, g)$ [why?]

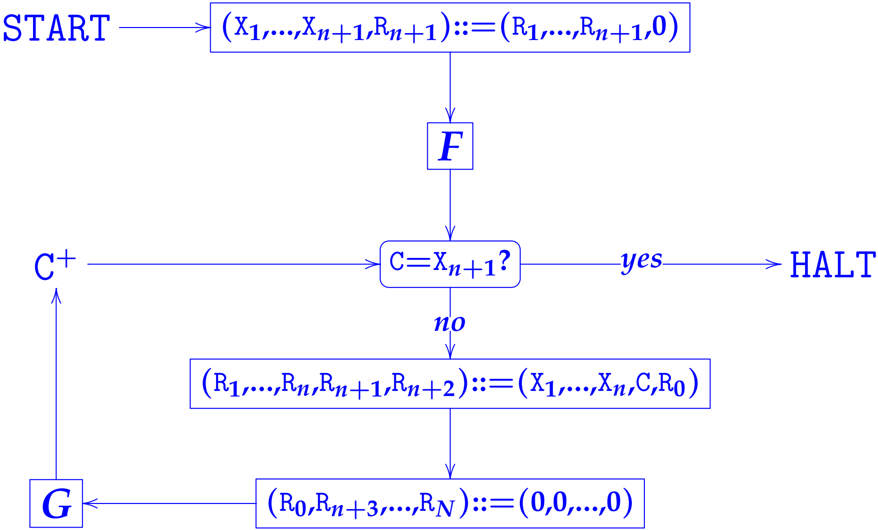
Definition. A [partial] function f is primitive recursive ($f \in \mathbf{PRIM}$) if it can be built up in finitely many steps from the basic functions by use of the operations of composition and primitive recursion.

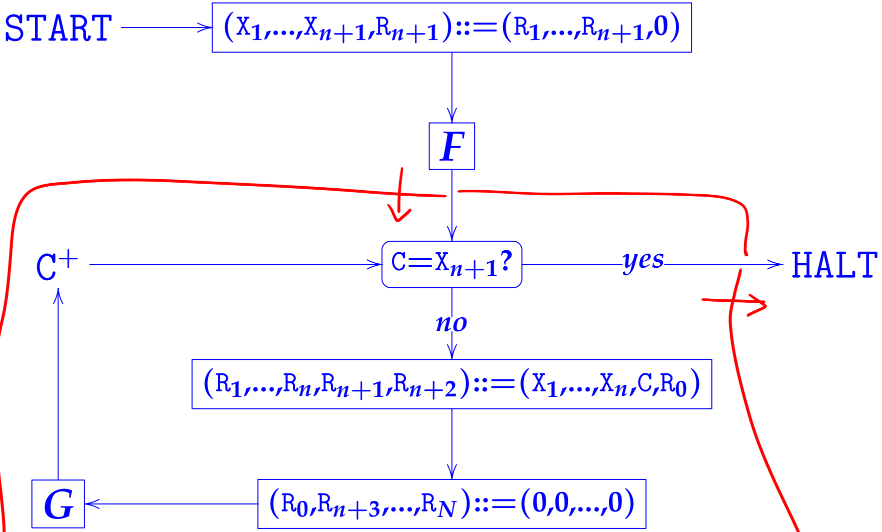
Theorem. Every $f \in \mathbf{PRIM}$ is computable.

Proof. Already proved: basic functions are computable; composition preserves computability. So just have to show:

$\rho^n(f,g) \in \mathbb{N}^{n+1} \rightharpoonup \mathbb{N}$ computable if $f \in \mathbb{N}^n \rightharpoonup \mathbb{N}$ and $g \in \mathbb{N}^{n+1} \rightharpoonup \mathbb{N}$ are.

Suppose f and g are computed by RM programs F and G (with our usual I/O conventions). Then the RM specified on the next slide computes $\rho^n(f,g)$. (We assume $\mathtt{X}_1, \ldots, \mathtt{X}_{n+1}, \mathtt{C}$ are some registers not mentioned in F and G; and that the latter only use registers $\mathtt{R}_0, \ldots, \mathtt{R}_N$, where $N \geq n+2$.)


START
$(X_1,\ldots,X_{n+1},R_{n+1}) ::= (R_1,\ldots,R_{n+1},0)$
F
$C = X_{n+1}$?
yes
HALT
no
$(R_1,\ldots,R_n,R_{n+1},R_{n+2}) ::= (X_1,\ldots,X_n,C,R_0)$
$(R_0,R_{n+3},\ldots,R_N) ::= (0,0,\ldots,0)$
G
C^+



while $C < X_{n+1}$ do
$(R_0, C) := (g(X_1, \ldots, X_n, C, R_0), C+1)$