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

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 $x = 0$

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

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.

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

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

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

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.

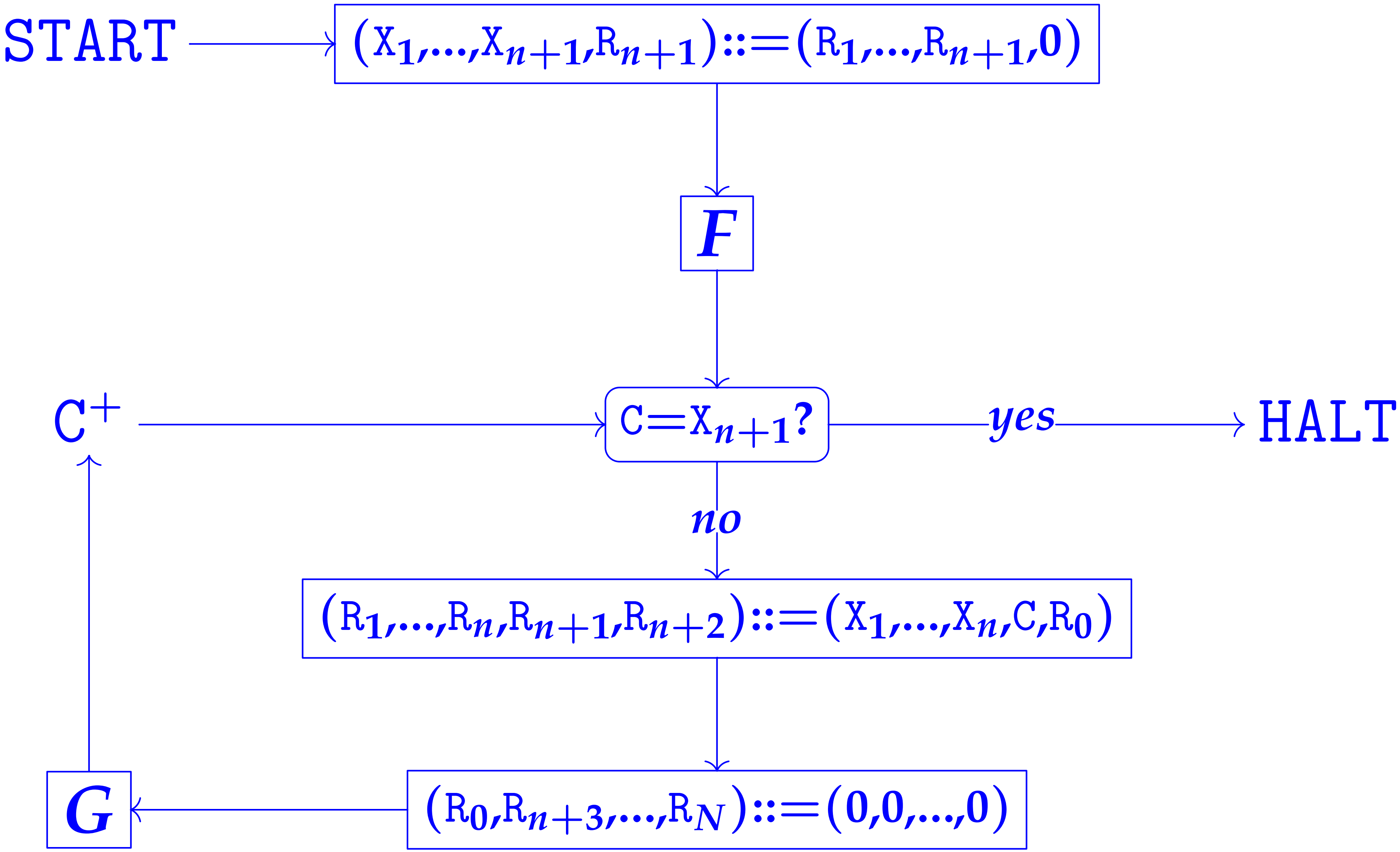
Definition. A [partial] function f is primitive recursive ($f \in \mathbf{PRIM}$) if it can be built up in finitely many steps from the basic functions by use of the operations of composition and primitive recursion.

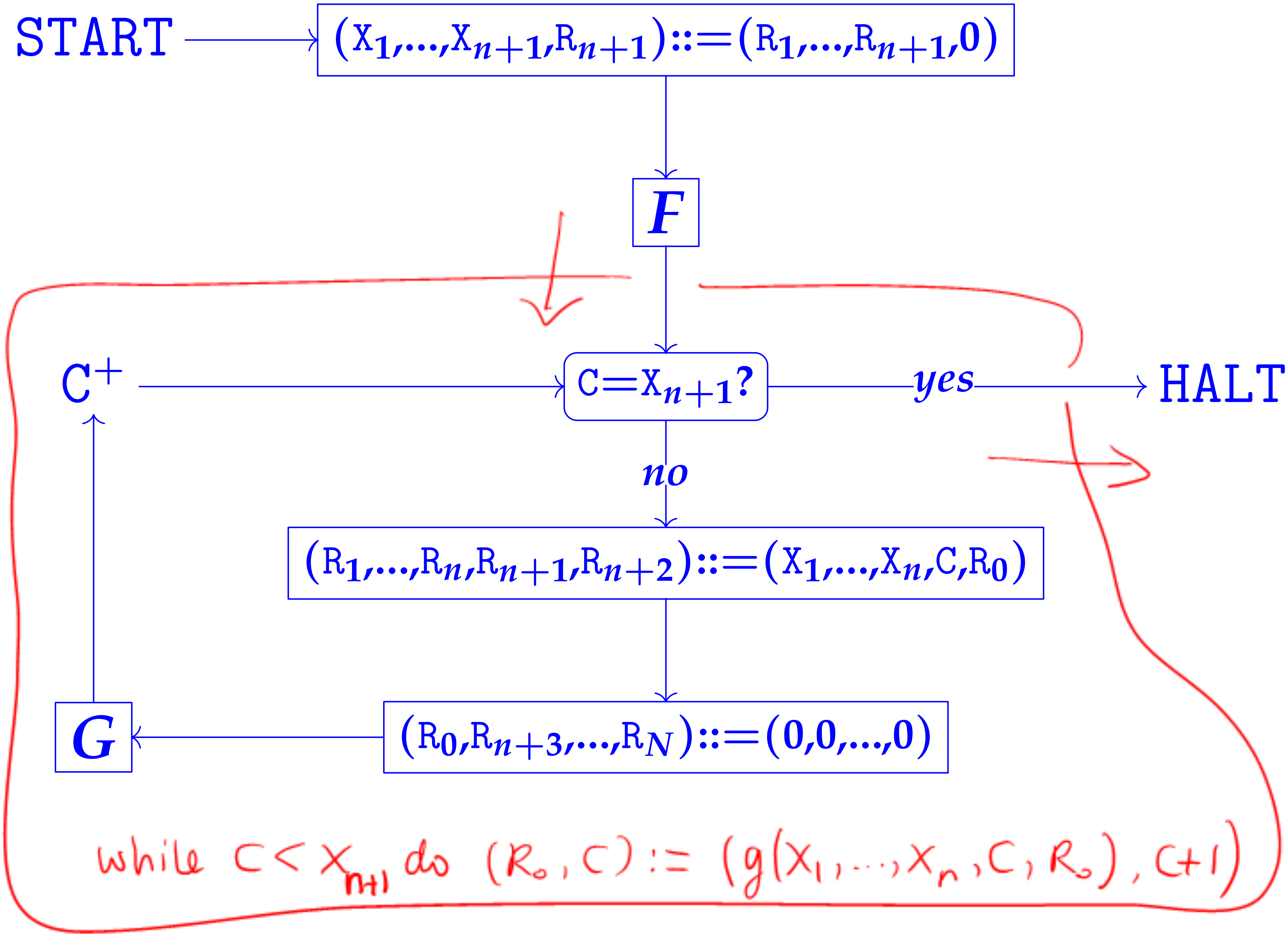
Theorem. Every $f \in \mathbf{PRIM}$ is computable.

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 $g \in \mathbb{N}^{n+2} \rightharpoonup \mathbb{N}$ 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
(X1,...,Xn+1,Rn+1)::=(R1,...,Rn+1,0)
F
C+
C=Xn+1?
yes
HALT
no
(R1,...,Rn,Rn+1,Rn+2)::=(X1,...,Xn,C,R0)
G
(R0,Rn+3,...,RN)::=(0,0,...,0)


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
C^+
$(R_1,\ldots,R_n,R_{n+1},R_{n+2}) ::= (X_1,\ldots,X_n,C,R_0)$
$(R_0,R_{n+3},\ldots,R_N) ::= (0,0,\ldots,0)$
G
while $C < X_{n+1}$ do $(R_0, C) := (g(X_1,\ldots,X_n,C,R_0), C+1)$

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?]

So we need something more in order to characterise all computable partial 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.

Minimization

Given a partial function $f \in \mathbb{N}^{n+1} \rightharpoonup \mathbb{N}$, define $\mu^n f \in \mathbb{N}^n \rightharpoonup \mathbb{N}$ by

$\mu^n f(\vec{x}) \triangleq$ least x such that $f(\vec{x}, x) = 0$ and for each $i = 0, \ldots, x-1$, $f(\vec{x}, i)$ is defined and > 0 (undefined if there is no such x)

In other words

$$\mu^n f = \{(\vec{x}, x) \in \mathbb{N}^{n+1} \mid \exists y_0, \ldots, y_x \; (\bigwedge_{i=0}^{x} f(\vec{x}, i) = y_i) \wedge (\bigwedge_{i=0}^{x-1} y_i > 0) \wedge y_x = 0\}$$

Example of minimization

integer part of $x_1/x_2 \equiv$ least x_3 such that

(undefined if $x_2 = 0$) $\quad x_1 < x_2(x_3 + 1)$

Example of minimization

$$\text{integer part of } x_1/x_2 \equiv \text{least } x_3 \text{ such that } x_1 < x_2(x_3+1)$$

(undefined if $x_2 = 0$)

$$\equiv \mu^2 f(x_1, x_2)$$

where $f \in \mathbb{N}^3 \to \mathbb{N}$ is

$$f(x_1, x_2, x_3) \triangleq \begin{cases} 1 & \text{if } x_1 \geq x_2(x_3+1) \\ 0 & \text{if } x_1 < x_2(x_3+1) \end{cases}$$

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

In other words, the set $\mathbf{PR}$ of partial 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, primitive recursion and minimization.

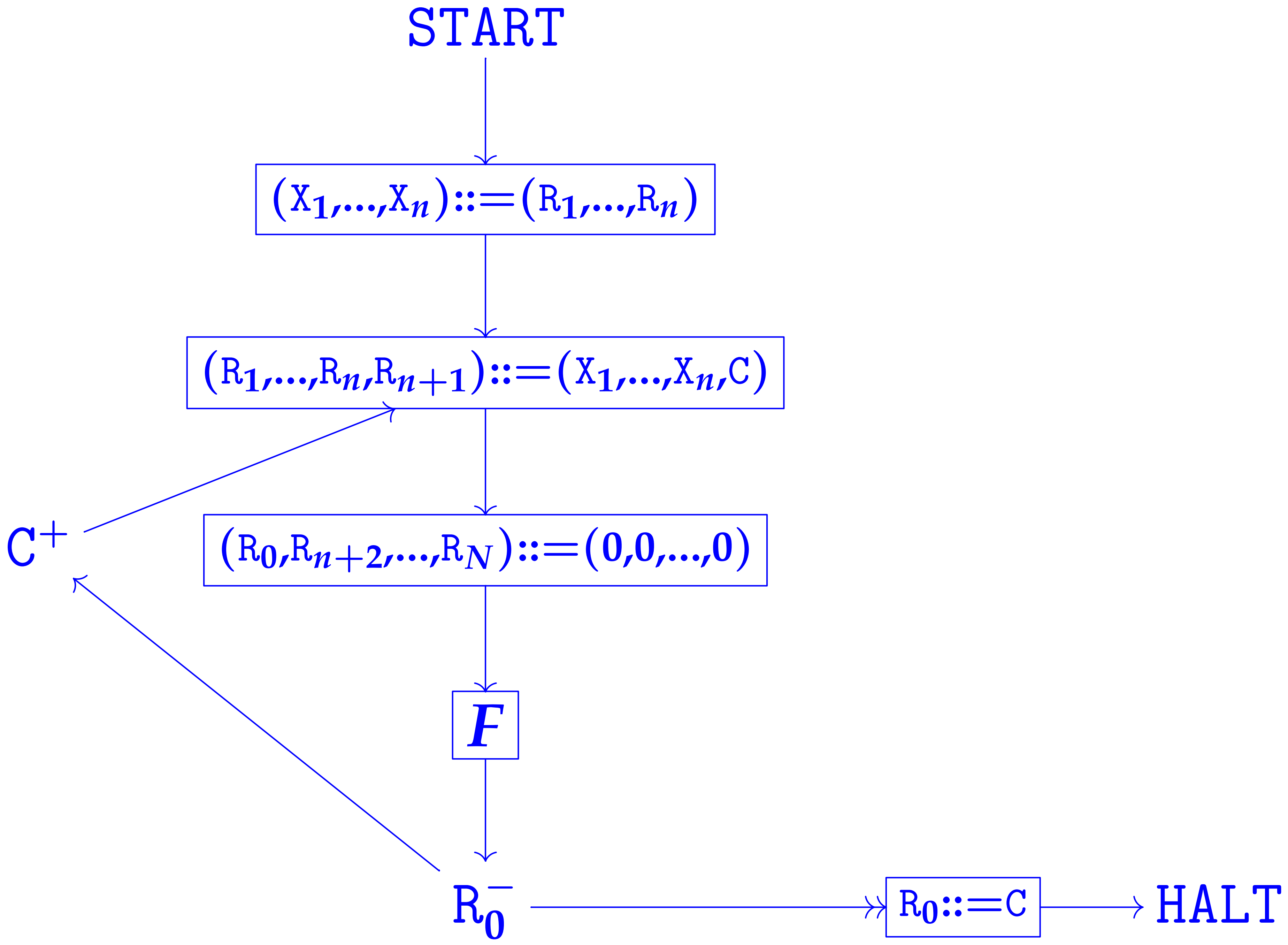
Definition. A partial function f is partial recursive ($f \in \mathbf{PR}$) if it can be built up in finitely many steps from the basic functions by use of the operations of composition, primitive recursion and minimization.

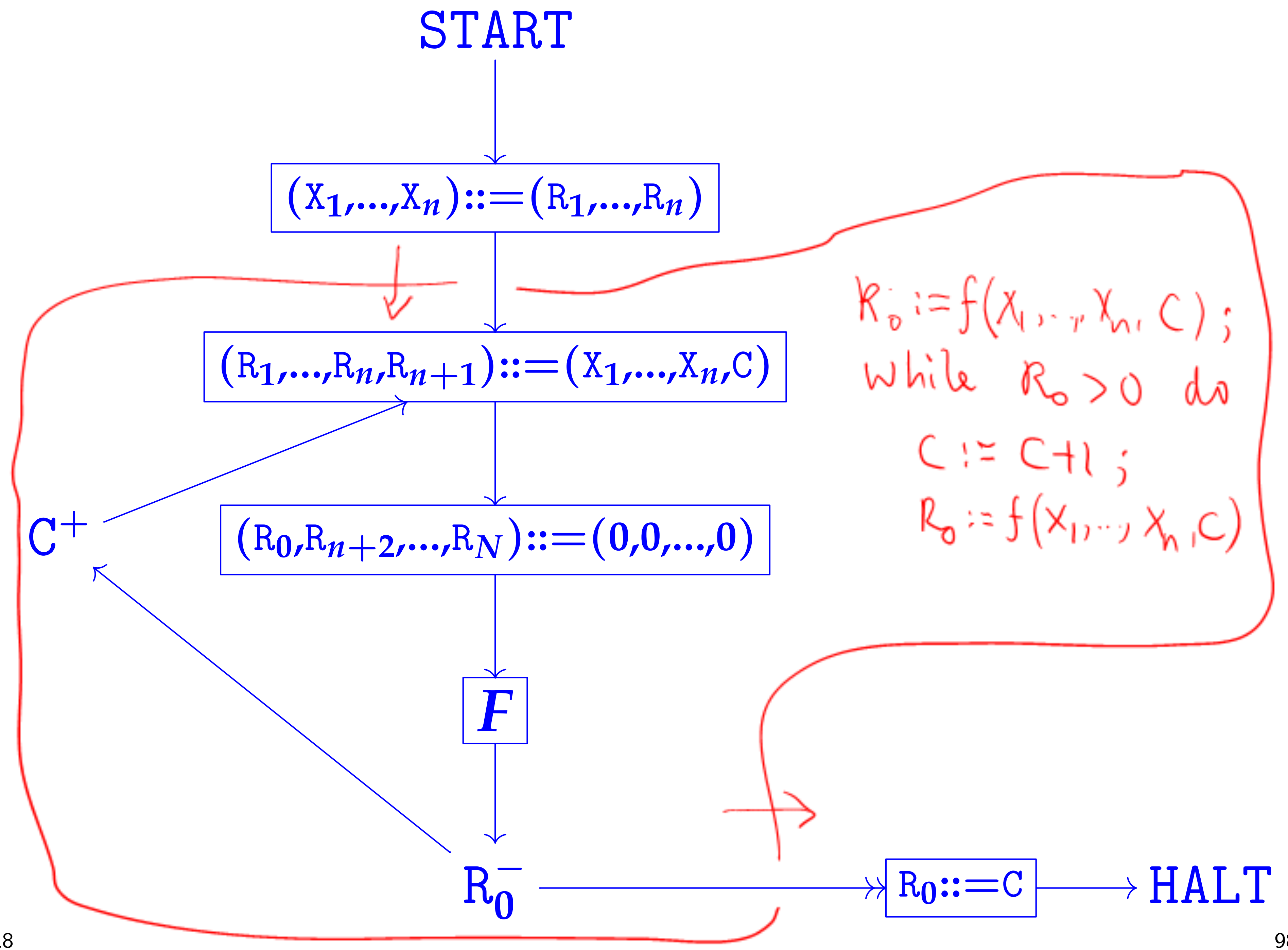
Theorem. Every $f \in \mathbf{PR}$ is computable.

Proof. Just have to show:

$\mu^n f \in \mathbb{N}^n \rightharpoonup \mathbb{N}$ is computable if $f \in \mathbb{N}^{n+1} \rightharpoonup \mathbb{N}$ is.

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


START
$(X_1,\ldots,X_n)::=(R_1,\ldots,R_n)$
$(R_1,\ldots,R_n,R_{n+1})::=(X_1,\ldots,X_n,C)$
$(R_0,R_{n+2},\ldots,R_N)::=(0,0,\ldots,0)$
F
C^+
R_0^-
$R_0::=C$
HALT


START
(X1,...,Xn)::=(R1,...,Rn)
(R1,...,Rn,Rn+1)::=(X1,...,Xn,C)
C+
(R0,Rn+2,...,RN)::=(0,0,...,0)
F
R0−
R0::=C
HALT
R0 := f(X1,...,Xn,C);
while R0 > 0 do
C := C+1;
R0 := f(X1,...,Xn,C)

Computable = partial recursive

Theorem. Not only is every $f \in \mathbf{PR}$ computable, but conversely, every computable partial function is partial recursive.

Proof (sketch). Let $f \in \mathbb{N}^n \rightharpoonup \mathbb{N}$ be computed by RM M with $N \geq n$ registers, say. Recall how we coded instantaneous configurations $c = (\ell, r_0, \ldots, r_N)$ of M as numbers $\ulcorner[\ell, r_0, \ldots, r_N]\urcorner$. It is possible to construct primitive recursive functions $lab, val_0, next_M \in \mathbb{N} \to \mathbb{N}$ satisfying

$$lab(\ulcorner[\ell, r_0, \ldots, r_N]\urcorner) = \ell$$
$$val_0(\ulcorner[\ell, r_0, \ldots, r_N]\urcorner) = r_0$$
$$next_M(\ulcorner[\ell, r_0, \ldots, r_N]\urcorner) = \text{code of } M\text{'s next configuration}$$

(Showing that $next_M \in \mathbf{PRIM}$ is tricky—proof omitted.)

(see Cutland, p95 et seq)

Proof sketch, cont.

Writing $\vec{x}$ for $x_1, \ldots, x_n$, let $config_M(\vec{x}, t)$ be the code of M's configuration after t steps, starting with initial register values $\mathtt{R}_0 = 0, \mathtt{R}_1 = x_1, \ldots, \mathtt{R}_n = x_n, \mathtt{R}_{n+1} = 0, \ldots, \mathtt{R}_N = 0$. It's in **PRIM** because:

$$\begin{cases} config_M(\vec{x}, 0) & = \ulcorner [0, 0, \vec{x}, \vec{0}] \urcorner \\ config_M(\vec{x}, t+1) & = next_M(config_M(\vec{x}, t)) \end{cases}$$

Proof sketch, cont.

Writing $\vec{x}$ for $x_1, \ldots, x_n$, let $config_M(\vec{x}, t)$ be the code of M's configuration after t steps, starting with initial register values $\mathtt{R}_0 = 0, \mathtt{R}_1 = x_1, \ldots, \mathtt{R}_n = x_n, \mathtt{R}_{n+1} = 0, \ldots, \mathtt{R}_N = 0$. It's in $\mathbf{PRIM}$ because:

$$\begin{cases} config_M(\vec{x}, 0) & = \ulcorner [0, 0, \vec{x}, \vec{0}] \urcorner \\ config_M(\vec{x}, t+1) & = next_M(config_M(\vec{x}, t)) \end{cases}$$

Can assume M has a single $\mathtt{HALT}$ as last instruction, Ith say (and no erroneous halts). Let $halt_M(\vec{x})$ be the number of steps M takes to halt when started with initial register values $\vec{x}$ (undefined if M does not halt). It satisfies

$$halt_M(\vec{x}) \equiv \text{least } t \text{ such that } I - lab(config_M(\vec{x}, t)) = 0$$

and hence is in $\mathbf{PR}$ (because $lab, config_M, I - (\ ) \in \mathbf{PRIM}$).

Proof sketch, cont.

Writing $\vec{x}$ for $x_1, \ldots, x_n$, let $config_M(\vec{x}, t)$ be the code of M's configuration after t steps, starting with initial register values $\mathtt{R}_0 = 0, \mathtt{R}_1 = x_1, \ldots, \mathtt{R}_n = x_n, \mathtt{R}_{n+1} = 0, \ldots, \mathtt{R}_N = 0$. It's in $\mathbf{PRIM}$ because:

$$\begin{cases} config_M(\vec{x}, 0) & = \ulcorner [0, 0, \vec{x}, \vec{0}] \urcorner \\ config_M(\vec{x}, t+1) & = next_M(config_M(\vec{x}, t)) \end{cases}$$

Can assume M has a single `HALT` as last instruction, Ith say (and no erroneous halts). Let $halt_M(\vec{x})$ be the number of steps M takes to halt when started with initial register values $\vec{x}$ (undefined if M does not halt). It satisfies

$$halt_M(\vec{x}) \equiv \text{least } t \text{ such that } I - lab(config_M(\vec{x}, t)) = 0$$

and hence is in $\mathbf{PR}$ (because $lab, config_M, I - (\ ) \in \mathbf{PRIM}$).

So $f \in \mathbf{PR}$, because $f(\vec{x}) \equiv val_0(config_M(\vec{x}, halt_M(\vec{x})))$.