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

\[
\begin{aligned}
  f_1(0) & \equiv 0 \\
  f_1(x + 1) & \equiv f_1(x) + (x + 1)
\end{aligned}
\]

\[f_1(x) = \text{sum of } 0, 1, 2, \ldots, x\]
Examples of recursive definitions

\[
\begin{align*}
    f_1(0) & \equiv 0 \\
    f_1(x + 1) & \equiv f_1(x) + (x + 1)
\end{align*}
\]

\[f_1(x) = \text{sum of } 0,1,2,\ldots,x\]

\[
\begin{align*}
    f_2(0) & \equiv 0 \\
    f_2(1) & \equiv 1 \\
    f_2(x + 2) & \equiv f_2(x) + f_2(x + 1)
\end{align*}
\]

\[f_2(x) = \text{xth Fibonacci number}\]
Examples of recursive definitions

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

\( f_1(x) = \text{sum of } 0, 1, 2, \ldots, x \)

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

\( f_2(x) = \text{xth Fibonacci number} \)

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

\( f_3(x) = \text{undefined except when } x = 0 \)
Examples of recursive definitions

\[
\begin{align*}
\begin{cases}
    f_1(0) & \equiv 0 \\
    f_1(x + 1) & \equiv f_1(x) + (x + 1)
\end{cases} \\
\end{align*}
\]

\[f_1(x) = \text{sum of } 0, 1, 2, \ldots, x\]

\[
\begin{align*}
\begin{cases}
    f_2(0) & \equiv 0 \\
    f_2(1) & \equiv 1 \\
    f_2(x + 2) & \equiv f_2(x) + f_2(x + 1)
\end{cases} \\
\end{align*}
\]

\[f_2(x) = \text{xth Fibonacci number}\]

\[
\begin{align*}
\begin{cases}
    f_3(0) & \equiv 0 \\
    f_3(x + 1) & \equiv f_3(x + 2) + 1
\end{cases} \\
\end{align*}
\]

\[f_3(x) \text{ undefined except when } x = 0\]

\[f_4(x) \equiv \begin{cases} 
    x - 10 & \text{if } x > 100 \\
    f_4(f_4(x + 11)) & \text{else}
\end{cases}\]

\[f_4 \text{ is McCarthy's "91 function", which maps } x \text{ to 91 if } x \leq 100 \text{ and to } x - 10 \text{ otherwise}\]
Examples of recursive definitions

\[
\begin{align*}
\left\{ \begin{array}{l}
f_1(0) \equiv 0 \\
f_1(x+1) \equiv f_1(x) + (x + 1)
\end{array} \right. \\
f_1(x) = \text{sum of } 0, 1, 2, \ldots, x
\end{align*}
\]

\[
\begin{align*}
\left\{ \begin{array}{l}
f_2(0) \equiv 0 \\
f_2(1) \equiv 1 \\
f_2(x+2) \equiv f_2(x) + f_2(x + 1)
\end{array} \right. \\
f_2(x) = x\text{th Fibonacci number}
\end{align*}
\]

\[
\begin{align*}
\left\{ \begin{array}{l}
f_3(0) \equiv 0 \\
f_3(x+1) \equiv f_3(x + 2) + 1
\end{array} \right. \\
f_3(x) \text{ undefined except when } x = 0
\end{align*}
\]

\[
f_4(x) \equiv \text{if } x > 100 \text{ then } x - 10 \\
\quad \text{else } f_4(f_4(x + 11)) \\
f_4 \text{ is McCarthy's "91 function", which maps } x \\
to 91 \text{ if } x \leq 100 \text{ and to } x - 10 \text{ otherwise}
\]
**Theorem.** Given $f \in \mathbb{N}^n \rightarrow \mathbb{N}$ and $g \in \mathbb{N}^{n+2} \rightarrow \mathbb{N}$, there is a unique $h \in \mathbb{N}^{n+1} \rightarrow \mathbb{N}$ satisfying

\[
\begin{align*}
    h(\vec{x}, 0) &\equiv f(\vec{x}) \\
    h(\vec{x}, x + 1) &\equiv g(\vec{x}, x, h(\vec{x}, x))
\end{align*}
\]

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**
Example: addition

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

\[
\begin{align*}
    add(x_1, 0) & \equiv x_1 \\
    add(x_1, x + 1) & \equiv add(x_1, x) + 1
\end{align*}
\]

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

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

Predecessor $\text{pred} \in \mathbb{N} \rightarrow \mathbb{N}$ satisfies:

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

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

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

Multiplication $\textit{mult} \in \mathbb{N}^2 \rightarrow \mathbb{N}$ satisfies:

\[
\begin{align*}
\text{mult}(x_1, 0) & \equiv 0 \\
\text{mult}(x_1, x + 1) & \equiv \text{mult}(x_1, x) + x_1
\end{align*}
\]

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

So $\textit{mult}$ can be built up from basic functions using composition and primitive recursion (since $\textit{add}$ can be).
Definition. A [partial] function \( f \) is primitive recursive \((f \in \text{PRIM})\) 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 \( \text{PRIM} \) 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 (\( f \in \text{PRIM} \)) 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 \( f \in \text{PRIM} \) is computable.

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

\[ \rho^n(f, g) \in \mathbb{N}^{n+1} \rightarrow \mathbb{N} \text{ computable if } f \in \mathbb{N}^n \rightarrow \mathbb{N} \text{ and } g \in \mathbb{N}^{n+2} \rightarrow \mathbb{N} \text{ are.} \]

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 \( x_1, \ldots, x_{n+1}, c \) are some registers not mentioned in \( F \) and \( G \); and that the latter only use registers \( R_0, \ldots, R_N \), where \( N \geq n + 2 \).)
\[
(x_1, \ldots, x_{n+1}, r_{n+1}) ::= (r_1, \ldots, r_n, 0)
\]

\[
F
\]

\[
C = x_{n+1}?
\]

- yes → HALT
- no

\[
(r_1, \ldots, r_n, r_{n+1}, r_{n+2}) ::= (x_1, \ldots, x_n, c, r_0)
\]

\[
G
\]

\[
(r_0, r_{n+3}, \ldots, r_N) ::= (0, 0, \ldots, 0)
\]
START \[ (x_1, \ldots, x_{n+1}, r_{n+1}) ::= (r_1, \ldots, r_{n+1}, 0) \]

\[ F \]

\[ C = x_{n+1}? \]

\( \text{yes} \) \rightarrow \text{HALT}

\( \text{no} \)

\[ (r_1, \ldots, r_n, r_{n+1}, r_{n+2}) ::= (x_1, \ldots, x_n, c, r_0) \]

\[ G \]

\[ (r_0, r_{n+3}, \ldots, r_N) ::= (0, 0, \ldots, 0) \]

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 (\( f \in \text{PRIM} \)) 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 \( f \in \text{PRIM} \) 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
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 \( f \in \mathbb{N}^{n+1} \rightarrow \mathbb{N} \), define \( \mu^n f \in \mathbb{N}^n \rightarrow \mathbb{N} \) by
\[
\mu^n f (\vec{x}) \triangleq \text{least } x \text{ such that } f(\vec{x}, x) = 0 \text{ and for each } i = 0, \ldots, x-1, \text{ } f(\vec{x}, i) \text{ is defined and } > 0 \text{ (undefined if there is no such } x)\]

In other words
\[
\mu^n f = \{ (\vec{x}, x) \in \mathbb{N}^{n+1} \mid \exists y_0, \ldots, y_x \ 
\left( \bigwedge_{i=0}^{x} f(\vec{x}, i) = y_i \right) \land \left( \bigwedge_{i=0}^{x-1} y_i > 0 \right) \land y_x = 0 \} \]
Example of minimization

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

(undefined if $x_2=0$) $x_1 < x_2(x_3 + 1)$
Example of minimization

integer part of \( x_1/x_2 \) \( \equiv \) least \( x_3 \) such that
\[
(\text{undefined if } x_2 = 0) \quad x_1 < x_2(x_3 + 1)
\]
\( \equiv \mu^2 f(x_1, x_2) \)

where \( f \in \mathbb{N}^3 \rightarrow \mathbb{N} \) 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** ($f \in \text{PR}$) 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 $\text{PR}$ 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** (\( f \in \text{PR} \)) 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 \( f \in \text{PR} \) is computable.

**Proof.** Just have to show:

\[ \mu^n f \in \mathbb{N}^n \rightarrow \mathbb{N} \text{ is computable if } f \in \mathbb{N}^{n+1} \rightarrow \mathbb{N} \text{ is.} \]

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 \( x_1, \ldots, x_n, c \) are some registers not mentioned in \( F \); and that the latter only uses registers \( R_0, \ldots, R_N \), where \( N \geq n + 1 \).)
\[ (X_1, \ldots, X_n) ::= (R_1, \ldots, R_n) \]

\[ (R_1, \ldots, R_n, R_{n+1}) ::= (X_1, \ldots, X_n, C) \]

\[ (R_0, R_{n+2}, \ldots, R_N) ::= (0, 0, \ldots, 0) \]

\[ F \]

\[ R_0^- \]

\[ R_0 ::= C \rightarrow \text{HALT} \]
\[(X_1, \ldots, X_n) ::= (R_1, \ldots, R_n)\]

\[(R_1, \ldots, R_n, R_{n+1}) ::= (X_1, \ldots, X_n, C)\]

\[(R_0, R_{n+2}, \ldots, R_N) ::= (0, 0, \ldots, 0)\]

\[R_0 ::= C\] \[\text{HALT}\]

\[R_0 ::= f(X_1, \ldots, X_n, C)\]

\[\text{while } R_0 > 0 \text{ do}\]

\[C ::= C + 1 ;\]

\[R_0 ::= f(X_1, \ldots, X_n, C)\]
Computable = partial recursive

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

**Proof (sketch).** Let $f \in \mathbb{N}^n \rightarrow \mathbb{N}$ 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 $\llbracket [\ell, r_0, \ldots, r_N] \rrbracket$. It is possible to construct primitive recursive functions $\text{lab}, \text{val}_0, \text{next}_M \in \mathbb{N} \rightarrow \mathbb{N}$ satisfying

$$
\text{lab}(\llbracket [\ell, r_0, \ldots, r_N] \rrbracket) = \ell
$$
$$
\text{val}_0(\llbracket [\ell, r_0, \ldots, r_N] \rrbracket) = r_0
$$
$$
\text{next}_M(\llbracket [\ell, r_0, \ldots, r_N] \rrbracket) = \text{code of } M\text{'s next configuration}
$$

(Showing that $\text{next}_M \in \text{PRIM}$ is tricky—proof omitted.)
Proof sketch, cont.

Writing $\vec{x}$ for $x_1, \ldots, x_n$, let $\text{config}_M(\vec{x}, t)$ be the code of $M$'s configuration after $t$ steps, starting with initial register values $R_0 = 0, R_1 = x_1, \ldots, R_n = x_n, R_{n+1} = 0, \ldots, R_N = 0$. It’s in PRIM because:

$$\begin{align*}
\text{config}_M(\vec{x}, 0) &= \llbracket 0, 0, \vec{x}, 0 \rrbracket \\
\text{config}_M(\vec{x}, t + 1) &= \text{next}_M(\text{config}_M(\vec{x}, t))
\end{align*}$$
Proof sketch, cont.

Writing $\vec{x}$ for $x_1, \ldots, x_n$, let $\text{config}_M(\vec{x}, t)$ be the code of $M$'s configuration after $t$ steps, starting with initial register values $R_0 = 0, R_1 = x_1, \ldots, R_n = x_n, R_{n+1} = 0, \ldots, R_N = 0$. It’s in PRIM because:

\[
\begin{align*}
\text{config}_M(\vec{x}, 0) &= \lceil [0, 0, \vec{x}, 0] \rceil \\
\text{config}_M(\vec{x}, t + 1) &= \text{next}_M(\text{config}_M(\vec{x}, t))
\end{align*}
\]

Can assume $M$ has a single HALT as last instruction, $I$ th say (and no erroneous halts). Let $\text{halt}_M(\vec{x})$ 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

\[
\text{halt}_M(\vec{x}) \equiv \text{least } t \text{ such that } I - \text{lab}(\text{config}_M(\vec{x}, t)) = 0
\]

and hence is in PR (because $\text{lab, config}_M, I - ( ) \in \text{PRIM}$).
Proof sketch, cont.

Writing $\vec{x}$ for $x_1, \ldots, x_n$, let $\text{config}_M(\vec{x}, t)$ be the code of $M$'s configuration after $t$ steps, starting with initial register values $R_0 = 0, R_1 = x_1, \ldots, R_n = x_n, R_{n+1} = 0, \ldots, R_N = 0$. It’s in PRIM because:

$$
\begin{align*}
\text{config}_M(\vec{x}, 0) &= \llbracket 0, 0, \vec{x}, 0 \rrbracket \\
\text{config}_M(\vec{x}, t + 1) &= \text{next}_M(\text{config}_M(\vec{x}, t))
\end{align*}
$$

Can assume $M$ has a single HALT as last instruction, $I$ th say (and no erroneous halts). Let $\text{halt}_M(\vec{x})$ 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

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

and hence is in PR (because $\text{lab}, \text{config}_M, I - (\quad) \in \text{PRIM}$).

So $f \in \text{PR}$, because $f(\vec{x}) \equiv \text{val}_0(\text{config}_M(\vec{x}, \text{halt}_M(\vec{x})))$. 