Computability Theory III

Primitive Recursive Function

Qingshui Xue

Shanghai Jiao Tong University

Sep. 28, 2015

Review Tips

Register

An Unlimited Register Machine (URM) has an infinite number of register labeled $R_1, R_2, R_3, \ldots$.

r_1	r_2	r_3	r_4	r_5	r_6	r_7	$\ldots$

$R_1 \quad R_2 \quad R_3 \quad R_4 \quad R_5 \quad R_6 \quad R_7 \quad \ldots$

Every register can hold a natural number at any moment.

The registers can be equivalently written as for example

$$[r_1, r_2, r_3]_1^3[r_4]_4^4[r_5, r_6, r_7]_5^7[0, 0, 0, \ldots]_8^\infty$$

or simply

$$[r_1, r_2, r_3]_1^3[r_4]_4^4[r_5, r_6, r_7]_5^7.$$

Instruction

Type	Instruction	Response of the URM
Zero	$Z(n)$	Replace r_n by 0.
Successor	$S(n)$	Add 1 to r_n.
Transfer	$T(m, n)$	Copy r_m to R_n.
Jump	$J(m, n, q)$	If $r_m = r_n$, go to the q-th instruction; otherwise go to the next instruction.

Recursive Function

Recursion Theory

Recursion Theory offers a mathematical model for the study of effective calculability.

1. All effective objects can be encoded by natural numbers.
2. All effective procedures can be modeled by functions from numbers to numbers.

Synopsis

1. Primitive Recursive Function

Primitive Recursive Function

Basic Definitions

Initial Function

1. The zero function
   - $\mathbf{0}$
   - $\mathbf{0}(\widetilde{x}) = 0$

Initial Function

1. The zero function
   - $\mathbf{0}$
   - $\mathbf{0}(\widetilde{x}) = 0$
2. The successor function
   - $s(x) = x + 1$

Initial Function

1. The zero function
   - $\mathbf{0}$
   - $\mathbf{0}(\widetilde{x}) = 0$
2. The successor function
   - $s(x) = x + 1$
3. The projection function
   - $U_i^n(x_1, \ldots, x_n) = x_i$

Composition

Suppose $f(y_1, \ldots, y_k)$ is a k-ary function and $g_1(\widetilde{x}), \ldots, g_k(\widetilde{x})$ are n-ary functions, where $\widetilde{x}$ abbreviates $x_1, \ldots, x_n$.

Composition

Suppose $f(y_1, \ldots, y_k)$ is a k-ary function and $g_1(\widetilde{x}), \ldots, g_k(\widetilde{x})$ are n-ary functions, where $\widetilde{x}$ abbreviates $x_1, \ldots, x_n$.

The composition function $h(\widetilde{x})$ is defined by

$$h(\widetilde{x}) \quad = \quad f(g_1(\widetilde{x}), \ldots, g_k(\widetilde{x})),$$

Recursion

Suppose that $f(\widetilde{x})$ is an n-ary function and $g(\widetilde{x}, y, z)$ is an $(n{+}2)$-ary function.

Recursion

Suppose that $f(\widetilde{x})$ is an n-ary function and $g(\widetilde{x}, y, z)$ is an $(n+2)$-ary function.

The recursion function $h(\widetilde{x}, y)$ is defined by

$$
\begin{aligned}
h(\widetilde{x}, 0) &= f(\widetilde{x}), && (1)\\
h(\widetilde{x}, y+1) &= g(\widetilde{x}, y, h(\widetilde{x}, y)). && (2)
\end{aligned}
$$

Recursion

Suppose that $f(\widetilde{x})$ is an n-ary function and $g(\widetilde{x}, y, z)$ is an $(n+2)$-ary function.

The recursion function $h(\widetilde{x}, y)$ is defined by

$$\begin{aligned} h(\widetilde{x}, 0) &= f(\widetilde{x}), && (1) \\ h(\widetilde{x}, y+1) &= g(\widetilde{x}, y, h(\widetilde{x}, y)). && (2) \end{aligned}$$

Clearly there is a unique function that satisfies (1) and (2).

Primitive Recursive Recursion

The set of primitive recursive function is the least set generated from the initial functions, composition and recursion.

Dummy Parameter

Proposition

Suppose that $f(y_1, \ldots, y_k)$ is a primitive recursive and that $x_{i_1}, \ldots, x_{i_k}$ is a sequence of k variables from $x_1, \ldots, x_n$ (possibly with repetition). Then the function h given by

$$h(x_1, \ldots, x_n) \quad = \quad f(x_{i_1}, \ldots, x_{i_k})$$

is primitive recursive.

Dummy Parameter

Proposition

Suppose that $f(y_1, \ldots, y_k)$ is a primitive recursive and that $x_{i_1}, \ldots, x_{i_k}$ is a sequence of k variables from $x_1, \ldots, x_n$ (possibly with repetition). Then the function h given by

$$h(x_1, \ldots, x_n) \quad = \quad f(x_{i_1}, \ldots, x_{i_k})$$

is primitive recursive.

Proof

$h(\widetilde{x}) = f(\mathrm{U}^n_{i_1}(\widetilde{x}), \ldots, \mathrm{U}^n_{i_k}(\widetilde{x}))$.

Basic Arithmetic Function

Basic Arithmetic Function

- $x + y$
- xy
- x^y

Basic Arithmetic Function

- $x + y$
- $$\begin{aligned} x + 0 &= x, \\ x + (y + 1) &= s(x + y). \end{aligned}$$
- xy
- x^y

Basic Arithmetic Function

- $x + y$
- $$\begin{aligned} x + 0 &= x, \\ x + (y + 1) &= s(x + y). \end{aligned}$$
- xy
- $$\begin{aligned} x0 &= 0, \\ x(y + 1) &= xy + x. \end{aligned}$$
- x^y

Basic Arithmetic Function

- $x + y$
- $$\begin{aligned} x + 0 &= x, \\ x + (y + 1) &= s(x + y). \end{aligned}$$
- xy
- $$\begin{aligned} x0 &= 0, \\ x(y + 1) &= xy + x. \end{aligned}$$
- x^y
- $$\begin{aligned} x^0 &= 1, \\ x^{y+1} &= x^y x. \end{aligned}$$

Quiz

$x + y + z$

Basic Arithmetic Function

- $x \dot{-} 1$

- $x \dot{-} y \stackrel{\text{def}}{=} \begin{cases} x - y, & \text{if } x \geq y, \\ 0, & \text{otherwise.} \end{cases}$

Basic Arithmetic Function

- $x \dot{-} 1$
- $$\begin{aligned} 0 \dot{-} 1 &= 0, \\ (x+1) \dot{-} 1 &= x. \end{aligned}$$
- $x \dot{-} y \stackrel{\text{def}}{=} \begin{cases} x - y, & \text{if } x \geq y, \\ 0, & \text{otherwise.} \end{cases}$

Basic Arithmetic Function

- $x \dot{-} 1$
-
$$\begin{aligned} 0 \dot{-} 1 &= 0, \\ (x+1) \dot{-} 1 &= x. \end{aligned}$$
- $x \dot{-} y \stackrel{\text{def}}{=} \begin{cases} x - y, & \text{if } x \geq y, \\ 0, & \text{otherwise.} \end{cases}$
-
$$\begin{aligned} x \dot{-} 0 &= x, \\ x \dot{-} (y+1) &= (x \dot{-} y) \dot{-} 1. \end{aligned}$$

Basic Arithmetic Function

- $\mathrm{sg}(x) \stackrel{\text{def}}{=} \begin{cases} 0, & \text{if } x = 0, \\ 1, & \text{if } x \neq 0. \end{cases}$

- $\overline{\mathrm{sg}}(x) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } x = 0, \\ 0, & \text{if } x \neq 0. \end{cases}$

Basic Arithmetic Function

- $\mathrm{sg}(x) \stackrel{\mathrm{def}}{=} \begin{cases} 0, & \text{if } x = 0, \\ 1, & \text{if } x \neq 0. \end{cases}$
- $$\begin{aligned} \mathrm{sg}(0) &= 0, \\ \mathrm{sg}(x+1) &= 1. \end{aligned}$$
- $\overline{\mathrm{sg}}(x) \stackrel{\mathrm{def}}{=} \begin{cases} 1, & \text{if } x = 0, \\ 0, & \text{if } x \neq 0. \end{cases}$

Basic Arithmetic Function

- $\text{sg}(x) \stackrel{\text{def}}{=} \begin{cases} 0, & \text{if } x = 0, \\ 1, & \text{if } x \neq 0. \end{cases}$
- $$\begin{aligned} \text{sg}(0) &= 0, \\ \text{sg}(x+1) &= 1. \end{aligned}$$
- $\overline{\text{sg}}(x) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } x = 0, \\ 0, & \text{if } x \neq 0. \end{cases}$
- $$\overline{\text{sg}}(x) = 1 \dot{-} \text{sg}(x).$$

Basic Arithmetic Function

- $|x - y|$
- $x!$
- $\min(x, y)$
- $\max(x, y)$

Basic Arithmetic Function

- $|x - y|$
- $|x - y| = (x \dot{-} y) + (y \dot{-} x)$
- $x!$

- $\min(x, y)$

- $\max(x, y)$

Basic Arithmetic Function

- $|x - y|$
- $|x - y| = (x \dot{-} y) + (y \dot{-} x)$
- $x!$
-
$$\begin{aligned} 0! &= 1, \\ (x+1)! &= x!(x+1). \end{aligned}$$
- $\min(x, y)$
- $\max(x, y)$

Basic Arithmetic Function

- $|x - y|$
- $|x - y| = (x \dot{-} y) + (y \dot{-} x)$
- $x!$
- $$\begin{aligned} 0! &= 1, \\ (x+1)! &= x!(x+1). \end{aligned}$$
- $\min(x, y)$
- $\min(x, y) = x \dot{-} (x \dot{-} y)$.
- $\max(x, y)$

Basic Arithmetic Function

- $|x - y|$
- $|x - y| = (x \dot{-} y) + (y \dot{-} x)$
- $x!$
- $$\begin{aligned} 0! &= 1, \\ (x+1)! &= x!(x+1). \end{aligned}$$
- $\min(x, y)$
- $\min(x, y) = x \dot{-} (x \dot{-} y)$.
- $\max(x, y)$
- $\max(x, y) = x + (y \dot{-} x)$.

Basic Arithmetic Function

$$\mathrm{rm}(x, y) \stackrel{\text{def}}{=} \text{the remainder when } y \text{ is devided by } x$$

$$\mathrm{rm}(x, y+1) \stackrel{\text{def}}{=} \begin{cases} \mathrm{rm}(x, y) + 1 & \text{if } \mathrm{rm}(x, y) + 1 < x, \\ 0, & \text{otherwise.} \end{cases}$$

Basic Arithmetic Function

$\mathrm{rm}(x, y) \stackrel{\text{def}}{=}$ the remainder when y is devided by x

$$\mathrm{rm}(x, y+1) \stackrel{\text{def}}{=} \begin{cases} \mathrm{rm}(x, y)+1 & \text{if } \mathrm{rm}(x, y)+1 < x, \\ 0, & \text{otherwise.} \end{cases}$$

The recursive definition is given by

$$\begin{aligned} \mathrm{rm}(x, 0) &= 0, \\ \mathrm{rm}(x, y+1) &= (\mathrm{rm}(x, y)+1)\mathrm{sg}(x \dot{-} (\mathrm{rm}(x, y)+1)). \end{aligned}$$

Basic Arithmetic Function

$\mathsf{qt}(x, y) \stackrel{\text{def}}{=}$ the quotient when y is devided by x

$$\mathsf{qt}(x, y+1) \stackrel{\text{def}}{=} \begin{cases} \mathsf{qt}(x, y) + 1, & \text{if } \mathsf{rm}(x, y) + 1 = x, \\ \mathsf{qt}(x, y), & \text{if } \mathsf{rm}(x, y) + 1 \neq x. \end{cases}$$

Basic Arithmetic Function

$\mathrm{qt}(x, y) \stackrel{\text{def}}{=}$ the quotient when y is devided by x

$$\mathrm{qt}(x, y+1) \stackrel{\text{def}}{=} \begin{cases} \mathrm{qt}(x, y) + 1, & \text{if } \mathrm{rm}(x, y) + 1 = x, \\ \mathrm{qt}(x, y), & \text{if } \mathrm{rm}(x, y) + 1 \neq x. \end{cases}$$

The recursive definition is given by

$$\begin{aligned} \mathrm{qt}(x, 0) &= 0, \\ \mathrm{qt}(x, y+1), &= \mathrm{qt}(x, y) + \overline{\mathrm{sg}}(x - (\mathrm{rm}(x, y) + 1)). \end{aligned}$$

Basic Arithmetic Function

$$\mathrm{div}(x, y) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } x \text{ divides } y, \\ 0, & \text{otherwise.} \end{cases}$$

Basic Arithmetic Function

$$\mathrm{div}(x, y) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } x \text{ divides } y, \\ 0, & \text{otherwise.} \end{cases}$$

$$\mathrm{div}(x, y) = \overline{\mathrm{sg}}(\mathrm{rm}(x, y)).$$

Bounded Minimalisation Operator

Bounded Sum and Bounded Product

Bounded sum:

$$\begin{aligned} \sum_{y<0} f(\widetilde{x}, y) &= 0, \\ \sum_{y<z+1} f(\widetilde{x}, y) &= \sum_{y<z} f(\widetilde{x}, y) + f(\widetilde{x}, z). \end{aligned}$$

Bounded product:

$$\begin{aligned} \prod_{y<0} f(\widetilde{x}, y) &= 1, \\ \prod_{y<z+1} f(\widetilde{x}, y) &= (\prod_{y<z} f(\widetilde{x}, y)) \cdot f(\widetilde{x}, z). \end{aligned}$$

Bounded Sum and Bounded Product

By composition the following functions are also primitive recursive if $k(\widetilde{x}, \widetilde{w})$ is primitive recursive:

$$\sum_{z<k(\widetilde{x},\widetilde{w})} f(\widetilde{x}, z)$$

and

$$\prod_{z<k(\widetilde{x},\widetilde{w})} f(\widetilde{x}, z).$$

Bounded Minimization Operator

Bounded search:

$$\mu z{<}y(f(\widetilde{x}, z) = 0) \stackrel{\text{def}}{=} \begin{cases} \text{the least } z < y, & \text{such that } f(\widetilde{x}, z) = 0; \\ y, & \text{if there is no such } z. \end{cases}$$

Bounded Minimization Operator

Bounded search:

$$\mu z{<}y(f(\widetilde{x}, z) = 0) \stackrel{\text{def}}{=} \begin{cases} \text{the least } z < y, & \text{such that } f(\widetilde{x}, z) = 0; \\ y, & \text{if there is no such } z. \end{cases}$$

Proposition

If $f(\widetilde{x}, z)$ is primitive recursive, then so is $\mu z{<}y(f(\widetilde{x}, z) = 0)$

Bounded Minimization Operator

Bounded search:

$$\mu z{<}y(f(\widetilde{x},z)=0) \stackrel{\text{def}}{=} \begin{cases} \text{the least } z < y, & \text{such that } f(\widetilde{x},z)=0; \\ y, & \text{if there is no such } z. \end{cases}$$

Proposition

If $f(\widetilde{x},z)$ is primitive recursive, then so is $\mu z{<}y(f(\widetilde{x},z)=0)$

Proof

$\mu z{<}y(f(\widetilde{x},z)=0) = \sum_{v<y}(\prod_{u<v+1} \mathrm{sg}(f(\widetilde{x},u)))$

Bounded Minimization Operator

If $f(\widetilde{x}, z)$ and $k(\widetilde{x}, \widetilde{w})$ are primitive recursive functions, then so is the function

$$\mu z{<}k(\widetilde{x}, \widetilde{w})(f(\widetilde{x}, z) = 0).$$

Primitive Recursive Predicate

Primitive Recursive Predicate

Suppose $M(x_1, \ldots, x_n)$ is an n-ary predicate of natural numbers. The characteristic function $c_M(\widetilde{x})$, where $\widetilde{x} = x_1, \ldots, x_n$, is

$$c_M(a_1, \ldots, a_n) = \begin{cases} 1, & \text{if } M(a_1, \ldots, a_n) \text{ holds,} \\ 0, & \text{if otherwise.} \end{cases}$$

The predicate $M(\widetilde{x})$ is primitive recursive if c_M is primitive recursive.

Closure Property

Proposition

The following statements are valid:

- If $R(\widetilde{x})$ is a primitive recursive predicate, then so is $\neg R(\widetilde{x})$.
- If $R(\widetilde{x})$, $S(\widetilde{x})$ are primitive recursive predicates, then the following predicates are primitive recursive:
  - $R(\widetilde{x}) \wedge S(\widetilde{x})$;
  - $R(\widetilde{x}) \vee S(\widetilde{x})$.
- If $R(\widetilde{x}, y)$ is a primitive recursive predicate, then the following predicates are primitive recursive:
  - $\forall z < y. R(\widetilde{x}, z)$;
  - $\exists z < y. R(\widetilde{x}, z)$.

Closure Property

Proposition

The following statements are valid:

- If $R(\widetilde{x})$ is a primitive recursive predicate, then so is $\neg R(\widetilde{x})$.
- If $R(\widetilde{x})$, $S(\widetilde{x})$ are primitive recursive predicates, then the following predicates are primitive recursive:
  - $R(\widetilde{x}) \wedge S(\widetilde{x})$;
  - $R(\widetilde{x}) \vee S(\widetilde{x})$.
- If $R(\widetilde{x}, y)$ is a primitive recursive predicate, then the following predicates are primitive recursive:
  - $\forall z < y.R(\widetilde{x}, z)$;
  - $\exists z < y.R(\widetilde{x}, z)$.

Proof

For example $c_{\forall z<y.R(\widetilde{x},z)}(\widetilde{x}, y) = \prod_{z<y} c_R(\widetilde{x}, z)$.

Definition by Case

Proposition

Suppose that $f_1(\widetilde{x}), \ldots, f_k(\widetilde{x})$ are primitive recursive functions, and $M_1(\widetilde{x}), \ldots, M_k(\widetilde{x})$ are primitive recursive predicates, such that for every $\widetilde{x}$ exactly one of $M_1(\widetilde{x}), \ldots, M_k(\widetilde{x})$ holds. Then the function $g(\widetilde{x})$ given by

$$g(\widetilde{x}) = \begin{cases} f_1(\widetilde{x}), & \text{if } M_1(\widetilde{x}) \text{ holds,} \\ f_2(\widetilde{x}), & \text{if } M_2(\widetilde{x}) \text{ holds,} \\ \vdots & \\ f_k(\widetilde{x}), & \text{if } M_k(\widetilde{x}) \text{ holds.} \end{cases}$$

is primitive recursive.

Definition by Case

Proposition

Suppose that $f_1(\widetilde{x}), \ldots, f_k(\widetilde{x})$ are primitive recursive functions, and $M_1(\widetilde{x}), \ldots, M_k(\widetilde{x})$ are primitive recursive predicates, such that for every $\widetilde{x}$ exactly one of $M_1(\widetilde{x}), \ldots, M_k(\widetilde{x})$ holds. Then the function $g(\widetilde{x})$ given by

$$g(\widetilde{x}) = \begin{cases} f_1(\widetilde{x}), & \text{if } M_1(\widetilde{x}) \text{ holds,} \\ f_2(\widetilde{x}), & \text{if } M_2(\widetilde{x}) \text{ holds,} \\ \vdots & \\ f_k(\widetilde{x}), & \text{if } M_k(\widetilde{x}) \text{ holds.} \end{cases}$$

is primitive recursive.

Proof

$g(\widetilde{x}) = c_{M_1}(\widetilde{x}) f_1(\widetilde{x}) + \ldots + c_{M_k}(\widetilde{x}) f_k(\widetilde{x})$

More Arithmetic Functions

More Arithmetic Functions

The following functions are primitive recursive.

1. $D(x) =$ the number of divisors of x;
2. $Pr(x) = \begin{cases} 1, & \text{if } x \text{ is prime,} \\ 0, & \text{if } x \text{ is not prime.} \end{cases}$
3. $\mathrm{p}_x =$ the x-th prime number;
4. $(x)_y = \begin{cases} k, & k \text{ is the exponent of } p_y \text{ in the prime} \\ & \text{factorisation of } x, \text{ for } x, y > 0, \\ 0, & \text{if } x = 0 \text{ or } y = 0. \end{cases}$

More Arithmetic Functions

Proof

1. $D(x) = \sum_{y<x+1} \mathrm{div}(y, x)$.

More Arithmetic Functions

Proof

1. $D(x) = \sum_{y<x+1} \mathrm{div}(y, x)$.
2. $Pr(x) = \overline{\mathrm{sg}}(|D(x) - 2|)$.

More Arithmetic Functions

Proof

1. $D(x) = \sum_{y<x+1} \mathrm{div}(y, x)$.
2. $Pr(x) = \overline{\mathrm{sg}}(|D(x) - 2|)$.
3. p_x can be recursively defined as follows:

$$
\begin{aligned}
\mathrm{p}_0 &= 0, \\
\mathrm{p}_{x+1} &= \mu z < (1 + \mathrm{p}_x!) \left(1 \dot{-} (z \dot{-} \mathrm{p}_x) Pr(z) = 0\right).
\end{aligned}
$$

More Arithmetic Functions

Proof

1. $D(x) = \sum_{y<x+1} \mathrm{div}(y, x)$.
2. $Pr(x) = \overline{\mathrm{sg}}(|D(x) - 2|)$.
3. p_x can be recursively defined as follows:

$$
\begin{aligned}
\mathrm{p}_0 &= 0, \\
\mathrm{p}_{x+1} &= \mu z < (1 + \mathrm{p}_x!) \left(1 \dot{-} (z \dot{-} \mathrm{p}_x) Pr(z) = 0\right).
\end{aligned}
$$

4. $(x)_y = \mu z{<}x(\mathrm{div}(\mathrm{p}_y^{z+1}, x) = 0)$.

Encoding a Finite Sequence

Suppose $s = (a_1, a_2, \ldots, a_n)$ is a finite sequence of numbers. It can be coded by the following number

$$b = \mathrm{p}_1^{a_1+1}\mathrm{p}_2^{a_2+1}\ldots\mathrm{p}_n^{a_n+1}.$$

Then the length of s can be recovered from

$$\mu z{<}b((b)_{z+1} = 0),$$

and the i-th component can be recovered from

$$(b)_i \dot{-} 1.$$

Not all Computable Functions are Primitive Recursive

Using the fact that all primitive recursive functions are total, a diagonalisation argument shows that non-primitive recursive computable functions must exist.

Not all Computable Functions are Primitive Recursive

Using the fact that all primitive recursive functions are total, a diagonalisation argument shows that non-primitive recursive computable functions must exist.

The same diagonalisation argument applies to all finite axiomatizations of computable total function.

Onward to the partial functions!