بسم الله الرحمن الرحیم

Theory of Computer Science

Second Part

Ali Shakiba
ali.shakiba@vru.ac.ir

Department of Computer Science
Vali-e-Asr University of Rafsanjan

Fall 2016

Composition

Let f be a function of k variables and let $g_1, \ldots, g_k$ be functions of n variables. Let

$$h(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n)).$$

The h is said to be obtained from f and $g_1, \ldots, g_k$ by *composition*.

Composition of (Partially) Computable Functions

Theorem 1.1. If h is obtained from the (partially) computable functions $f, g_1, \ldots, g_k$ by composition, then h is (partially) computable.

Proof. The following program computes h:

$$Z_1 \leftarrow g_1(X_1, \ldots, X_n)$$
$$\ldots$$
$$Z_k \leftarrow g_k(X_1, \ldots, X_n)$$
$$Y \leftarrow f(Z_1, \ldots, Z_k)$$

If $f, g_1, \ldots, g_k$ are total, so is h. $\square$

More Recursion

$$\begin{aligned} h(x_1, \ldots, x_n, 0) &= f(x_1, \ldots, x_n) \\ h(x_1, \ldots, x_n, t+1) &= g(t, h(x_1, \ldots, x_n, t), x_1, \ldots, x_n), \end{aligned}$$

where f is a total function of n variables, and g is a total function of $n+2$ variables. Function h of $n+1$ variable is said to be obtained from g by *primitive recursion*, or simply *recursion*, from f and g.

More Recursion of Computable Functions

Theorem 2.2. If h is obtained from g as in the previous slide and let g be computable. Then then h is also computable.

Proof. The following program computes $h(x_1, \ldots, x_n, x_{n+1})$:

$$
\begin{array}{ll}
 & Y \leftarrow f(x_1, \ldots, x_n) \\
[A] & \text{IF } X_{n+1} = 0 \text{ GOTO } E \\
 & Y \leftarrow g(Z, Y, X_1, \ldots, X_n) \\
 & Z \leftarrow Z + 1 \\
 & X_{n+1} \leftarrow X_{n+1} - 1 \\
 & \text{GOTO } A
\end{array}
$$

$\square$

Initial Functions

The following functions are called *initial functions*:

$$
\begin{aligned}
s(x) &= x + 1, \\
n(x) &= 0, \\
u_i^n(x_1, \ldots, x_n) &= x_i, \quad 1 \leq i \leq n.
\end{aligned}
$$

Note: Function u_i^n is called the *projection function*. For example, $u_3^4(x_1, x_2, x_3, x_4) = x_3$.

Primitive Recursively Closed (PRC)

A class of total functions $\mathscr{C}$ is called a *PRC* class if

- the initial functions belong to $\mathscr{C}$,
- a function obtained from functions belonging to $\mathscr{C}$ by either composition or recursion also belongs to $\mathscr{C}$.

Computable Functions are Primitive Recursively Closed

Theorem 3.1. The class of computable functions is a PRC class.

Proof. We have shown computable functions are closed under composition and recursion (Theorem 1.1 & 2.2). We need only verify the initial functions are computable. They are computed by the following programs.

$\boxed{s(x) = x + 1}$ $Y \leftarrow X + 1$;

$\boxed{n(x)}$ the empty program;

$\boxed{u_i^n(x_1, \ldots, x_n)}$ $Y \leftarrow X_i$.

□

Primitive Recursive Functions

A function is called *primitive recursive* if it can be obtained from the initial functions by a finite number of applications of composition and recursion.

Primitive Recursive Functions

A function is called *primitive recursive* if it can be obtained from the initial functions by a finite number of applications of composition and recursion.

Note that, by the above definition and the definition of Primitive Recursively Closed (PRC), it follows that:

Corollary 3.2. The class of primitive recursive function is a PRC class.

Primitive Recursive Functions & PRC Classes

Theorem 3.3. A function is primitive recursive if and only if it belongs to every PRC class.

Proof. ($\Leftarrow$) If a function belongs to every PRC class, then by Corollary 3.2, it belongs to the class of primitive recursive functions.

($\Rightarrow$) If f is primitive recursive, then there is a list of functions $f_1, f_2, \ldots, f_n$ such that $f_n = f$ and for each $f_i, 1 \leq i < n$, either

- f_i is an initial function, or
- f_i can be obtained from the preceding functions in the list by composition or recursion.

However, the initial functions belong to any PRC class $\mathscr{C}$. Furthermore, all functions obtained from functions in $\mathscr{C}$ by composition or recursion also belong to $\mathscr{C}$. It follows that each function $f_1, f_2, \ldots, f_n = f$ in the above list is in $\mathscr{C}$. □

Primitive Recursive Functions Are Computable

Corollary 3.4. Every primitive recursive function is computable.

Proof. By Theorem 3.4, every primitive recursive function belongs to the PRC class of computable functions so is computable. $\square$

Primitive Recursive Functions Are Computable

Corollary 3.4. Every primitive recursive function is computable.

Proof. By Theorem 3.4, every primitive recursive function belongs to the PRC class of computable functions so is computable. □

Note that,

- If a function f is shown to be primitive recursive, by the above Corollary, f can be expressed as a program in language $\mathscr{S}$.
- Not only we know there is program in $\mathscr{S}$ for f, by Theorem 3.1 (1.1 & 2.2), we also know how to write this program.
- Furthermore, the program so written will always terminate.

Primitive Recursive Functions Are Computable

Corollary 3.4. Every primitive recursive function is computable.

Proof. By Theorem 3.4, every primitive recursive function belongs to the PRC class of computable functions so is computable. □

Note that,

- If a function f is shown to be primitive recursive, by the above Corollary, f can be expressed as a program in language $\mathscr{S}$.
- Not only we know there is program in $\mathscr{S}$ for f, by Theorem 3.1 (1.1 & 2.2), we also know how to write this program.
- Furthermore, the program so written will always terminate.

However, if a function f is computable (that is, it is total and expressible in $\mathscr{S}$), it is not necessarily that f is primitive recursive. (A counter example will be shown later in this course.)

Function $f(x, y) = x + y$ Is Primitive Recursive

Function f can be defined by the recursion equations:

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

The above can be rewritten as

$$\begin{aligned} f(x, 0) &= u_1^1(x), \\ f(x, y+1) &= g(y, f(x, y), x), \end{aligned}$$

where

$$g(x_1, x_2, x_3) = s(u_2^3(x_1, x_2, x_3)).$$

Function $h(x, y) = x \cdot y$ Is Primitive Recursive

Function h can be defined by the recursion equations:

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

The above can be rewritten as

$$\begin{aligned} h(x, 0) &= n(x), \\ h(x, y+1) &= g(y, h(x, y), x), \end{aligned}$$

where

$$\begin{aligned} g(x_1, x_2, x_3) &= f(u_2^3(x_1, x_2, x_3), u_3^3(x_1, x_2, x_3)), \\ f(x, y) &= x + y. \end{aligned}$$

Function $h(x) = x!$ Is Primitive Recursive

Function $h(x)$ can be defined by

$$\begin{aligned} h(0) &= 1, \\ h(t+1) &= g(t, h(t)), \end{aligned}$$

where

$$g(x_1, x_2) = s(x_1) \cdot x_2.$$

Note that g is primitive recursive because

$$g(x_1, x_2) = s(u_1^2(x_1, x_2)) \cdot u_2^2(x_1, x_2).$$

Function $power(x, y) = x^y$ Is Primitive Recursive

Function *power* can be defined by

$$\begin{aligned} power(x, 0) &= 1, \\ power(x, y+1) &= power(x, y) \cdot x. \end{aligned}$$

Note that these equations assign the value 1 to the "indeterminate" 0^0.

The above definition can be further rewritten into ....

The Predecessor Function Is Primitive Recursive

The predecessor function $pred(x)$ is defined as follows:

$$pred(x) = \begin{cases} x - 1 & \text{if } x \neq 0 \\ 0 & \text{if } x = 0. \end{cases}$$

Note that function $pred$ corresponds to the instruction $X \leftarrow X - 1$ in programming language $\mathscr{S}$.

The above definition can be further rewritten into . . . .

Function $x \dot{-} y$ Is Primitive Recursive

Function $x \dot{-} y$ is defined as follows:

$$x \dot{-} y = \begin{cases} x - y & \text{if } x \geq y \\ 0 & \text{if } x < y. \end{cases}$$

Note that function $x \dot{-} y$ is different from function $x - y$, which is undefined if $x < y$. In particular, $x \dot{-} y$ is total while $x - y$ is not.

Function $x \dot{-} y$ Is Primitive Recursive

Function $x \dot{-} y$ is defined as follows:

$$x \dot{-} y = \begin{cases} x - y & \text{if } x \geq y \\ 0 & \text{if } x < y. \end{cases}$$

Note that function $x \dot{-} y$ is different from function $x - y$, which is undefined if $x < y$. In particular, $x \dot{-} y$ is total while $x - y$ is not.

Function $x \dot{-} y$ is primitive recursive because

$$\begin{aligned} x \dot{-} 0 &= x, \\ x \dot{-} (t+1) &= pred(x \dot{-} t). \end{aligned}$$

The above definition can be further rewritten into . . . .

Function $|x - y|$ Is Primitive Recursive

Function $|x - y|$ can be defined as follows:

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

Function $|x - y|$ Is Primitive Recursive

Function $|x - y|$ can be defined as follows:

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

It is primitive recursive because the above definition can be further rewritten into . . . .

Is Function $\alpha(x)$ below Primitive Recursive?

Function $\alpha(x)$ is defined as:

$$\alpha(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{if } x \neq 0. \end{cases}$$

Is Function $\alpha(x)$ below Primitive Recursive?

Function $\alpha(x)$ is defined as:

$$\alpha(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{if } x \neq 0. \end{cases}$$

It is primitive recursive because ....

$x = y$ Is Primitive Recursive

Is the function $d(x, y)$ below primitive recursive?

$$d(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases}$$

$x = y$ Is Primitive Recursive

Is the function $d(x, y)$ below primitive recursive?

$$d(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases}$$

It is because $d(x, y) = \alpha(|x - y|)$.

Is $x \leq y$ Primitive Recursive?

Is $x \leq y$ Primitive Recursive?

It is primitive recursive because $x \leq y = \alpha(x \dot{-} y)$.

Logic Connectives Are Primitive Recursively Closed

Theorem 5.1. Let $\mathscr{C}$ be a PRC class. If P, Q are predicates that belong to $\mathscr{C}$, then so are $\sim P$, $P \vee Q$, and $P \& Q$.

Logic Connectives Are Primitive Recursively Closed

Theorem 5.1. Let $\mathscr{C}$ be a PRC class. If P, Q are predicates that belong to $\mathscr{C}$, then so are $\sim P$, $P \vee Q$, and $P \& Q$.

Proof. We define $\sim P$, $P \vee Q$, and $P \& Q$ as follows:

$$\begin{aligned} \sim P &= \alpha(P) \\ P \,\&\, Q &= P \cdot Q \\ P \vee Q &= \sim(\sim P \,\&\, \sim Q) \end{aligned}$$

Logic Connectives Are Primitive Recursively Closed

Theorem 5.1. Let $\mathscr{C}$ be a PRC class. If P, Q are predicates that belong to $\mathscr{C}$, then so are $\sim P$, $P \vee Q$, and $P \& Q$.

Proof. We define $\sim P$, $P \vee Q$, and $P \& Q$ as follows:

$$\begin{aligned} \sim P &= \alpha(P) \\ P \,\&\, Q &= P \cdot Q \\ P \vee Q &= \sim(\sim P \,\&\, \sim Q) \end{aligned}$$

We conclude that $\sim P$, $P \vee Q$, and $P \& Q$ all belong to $\mathscr{C}$. $\square$

Logic Connectives Are Primitive Recursive and Computable

Corollary 5.2. If P, Q are primitive recursive predicates, then so are $\sim P$, $P \vee Q$, and $P \& Q$.

Logic Connectives Are Primitive Recursive and Computable

Corollary 5.2. If P, Q are primitive recursive predicates, then so are $\sim P$, $P \vee Q$, and $P \& Q$.

Corollary 5.3. If P, Q are computable predicates, then so are $\sim P$, $P \vee Q$, and $P \& Q$.

Is $x < y$ Primitive Recursive?

Is $x < y$ Primitive Recursive?

It is primitive recursive because

$$x < y \;\Leftrightarrow\; \sim (y \leq x).$$

Definition by Cases

Theorem 5.4. Let $\mathscr{C}$ be a PRC class. Let functions g, h and predicate P belong to $\mathscr{C}$. Let function

$$f(x_1, \ldots, x_n) = \begin{cases} g(x_1, \ldots, x_n) & \text{if } P(x_1, \ldots, x_n) \\ h(x_1, \ldots, x_n) & \text{otherwise.} \end{cases}$$

Then f belongs to $\mathscr{C}$.

Definition by Cases

Theorem 5.4. Let $\mathscr{C}$ be a PRC class. Let functions g, h and predicate P belong to $\mathscr{C}$. Let function

$$f(x_1,\ldots,x_n) = \begin{cases} g(x_1,\ldots,x_n) & \text{if } P(x_1,\ldots,x_n) \\ h(x_1,\ldots,x_n) & \text{otherwise.} \end{cases}$$

Then f belongs to $\mathscr{C}$.

Proof. Function f belongs to $\mathscr{C}$ because

$$\begin{aligned} f(x_1,\ldots,x_n) &= g(x_1,\ldots,x_n)\cdot P(x_1,\ldots,x_n) \\ &+ h(x_1,\ldots,x_n)\cdot \alpha(P(x_1,\ldots,x_n)). \end{aligned}$$

□

Definition by Cases, More

Corollary 5.5. Let $\mathscr{C}$ be a PRC class. Let n-ary functions $g_1, \ldots, g_m, h$ and predicates $P_1, \ldots, P_m$ belong to $\mathscr{C}$, and let

$$P_i(x_1, \ldots, x_n) \mathbin{\&} P_j(x_1, \ldots, x_n) \;=\; 0$$

for all $1 \leq i \leq j \leq m$ and all $x_1, \ldots, x_n$. If

$$f(x_1, \ldots, x_n) = \begin{cases} g_1(x_1, \ldots, x_n) & \text{if } P_1(x_1, \ldots, x_n) \\ \vdots & \vdots \\ g_m(x_1, \ldots, x_n) & \text{if } P_m(x_1, \ldots, x_n) \\ h(x_1, \ldots, x_n) & \text{otherwise.} \end{cases}$$

then f also belongs to $\mathscr{C}$.

Definition by Cases, More

Corollary 5.5. Let $\mathscr{C}$ be a PRC class. Let n-ary functions $g_1, \ldots, g_m, h$ and predicates $P_1, \ldots, P_m$ belong to $\mathscr{C}$, and let

$$P_i(x_1, \ldots, x_n) \; \& \; P_j(x_1, \ldots, x_n) \;=\; 0$$

for all $1 \leq i \leq j \leq m$ and all $x_1, \ldots, x_n$. If

$$f(x_1, \ldots, x_n) = \begin{cases} g_1(x_1, \ldots, x_n) & \text{if } P_1(x_1, \ldots, x_n) \\ \vdots & \vdots \\ g_m(x_1, \ldots, x_n) & \text{if } P_m(x_1, \ldots, x_n) \\ h(x_1, \ldots, x_n) & \text{otherwise.} \end{cases}$$

then f also belongs to $\mathscr{C}$.

Proof. Proved by a mathematical induction on m. □

Iterated Operations

Theorem 6.1. Let $\mathscr{C}$ be a PRC class. If function $f(t, x_1, \ldots, x_n)$ belongs to $\mathscr{C}$, then so do the functions g and h

$$g(y, x_1, \ldots, x_n) = \sum_{t=0}^{y} f(t, x_1, \ldots, x_n)$$

$$h(y, x_1, \ldots, x_n) = \prod_{t=0}^{y} f(t, x_1, \ldots, x_n)$$

Iterated Operations

Theorem 6.1. Let $\mathscr{C}$ be a PRC class. If function $f(t, x_1, \ldots, x_n)$ belongs to $\mathscr{C}$, then so do the functions g and h

$$g(y, x_1, \ldots, x_n) = \sum_{t=0}^{y} f(t, x_1, \ldots, x_n)$$

$$h(y, x_1, \ldots, x_n) = \prod_{t=0}^{y} f(t, x_1, \ldots, x_n)$$

Proof. Functions g and h each can be recursively defined as

$$\begin{aligned}
g(0, x_1, \ldots, x_n) &= f(0, x_1, \ldots, x_n), \\
g(t+1, x_1, \ldots, x_n) &= g(t, x_1, \ldots, x_n) + f(t+1, x_1, \ldots, x_n), \\
h(0, x_1, \ldots, x_n) &= f(0, x_1, \ldots, x_n), \\
h(t+1, x_1, \ldots, x_n) &= h(t, x_1, \ldots, x_n) \cdot f(t+1, x_1, \ldots, x_n).
\end{aligned}$$

Iterated Operations, More

Corollary 6.2. Let $\mathscr{C}$ be a PRC class. If function $f(t, x_1, \ldots, x_n)$ belongs to $\mathscr{C}$, then so do the functions

$$g(y, x_1, \ldots, x_n) = \sum_{t=1}^{y} f(t, x_1, \ldots, x_n)$$

and

$$h(y, x_1, \ldots, x_n) = \prod_{t=1}^{y} f(t, x_1, \ldots, x_n).$$

In the above, we assume that

$$\begin{aligned} g(0, x_1, \ldots, x_n) &= 0, \\ h(0, x_1, \ldots, x_n) &= 1. \end{aligned}$$