Computer Science Theory

(Master Course)

Chapter 3:

Primitive Recursive Functions

Mohammad Farshi
Department of Computer Science
Yazd University

1395-1

Composition

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

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

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

- **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 obviously computes h:

$$\begin{array}{rcl} Z_1 & \leftarrow & g_1(X_1, \ldots, X_n) \\ & \vdots & \\ Z_k & \leftarrow & g_k(X_1, \ldots, X_n) \\ Y & \leftarrow & f(Z_1, \ldots, Z_k) \end{array}$$

- **Definition:** Let
  g: a total function of two variables
  k: a fixed number
  Then h is said to be obtained from g by primitive recursion, or simply recursion if

$$h(0) = k,$$
$$h(t+1) = g(t, h(t)).$$

- **Theorem 2.1.** If g is computable, then h is also computable.

Theorem 2.1. If g is computable, then h is computable.

Proof.

- The constant function $f(x) = k$ is computable (by a program with k statement $Y \leftarrow Y + 1$). So we have macro $Y \leftarrow k$.
- The following program computes h:

$$
\begin{array}{ll}
 & Y \leftarrow k \\
[A] & \text{IF } X = 0 \text{ GOTO } E \\
 & Y \leftarrow g(Z, Y) \\
 & Z \leftarrow Z + 1 \\
 & X \leftarrow X - 1 \\
 & \text{GOTO } A
\end{array}
$$

□

- **Definition:** Let
  f: a total function of n variables
  g: a total function of $n+2$ variables
  Then h is said to be obtained from g by primitive recursion, or simply recursion if

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

- **Theorem 2.1.** If g is computable, then h is also computable.

Recursion

Theorem 2.1. If g is computable, then h is computable.

Proof.

- The following program computes h:

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

□

PRC Classes

Primitive Recursively Closed

Initial Function

- $s(x) = x + 1$
- $n(x) = 0$
- (projection functions) for each $1 \leq i \leq n$,
  $u_i^n(x_1, \ldots, x_n) = x_i$

A PRC class:

A class ϕ of total functions is called a PRC class if

- The initial functions belongs to ϕ,
- It is closed under composition and recursion.

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

Proof. By Theorems 1.1, 2.1, and 2.2, we need only verify that the initial functions are computable.

- $s(x) = x + 1$ is computed by $Y \leftarrow X + 1$.
- $n(x)$ is computed by the empty program.
- $u_i^n(x_1, \ldots, x_n)$ is computed by the program $Y \leftarrow X_i$.

Definition: primitive recursive function

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

Corollary 3.2.

The class of primitive recursive functions is a PRC class.

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

Proof. $(\Leftarrow)$ If f belongs to every PRC class, then, in particular, by Corollary 3.2, it belongs to the class of primitive recursive functions.
$(\Rightarrow)$ Let f be a primitive recursive function and let ϕ be some PRC class. We want to show that f belongs to ϕ. Since f is a primitive recursive function, there is a list $f_1, f_2, \ldots, f_n$ of functions such that $f_n = f$ and each f_i is either an initial function or can be obtained from preceding functions in the list by composition or recursion.
Now the initial functions certainly belong to the PRC class ϕ. Moreover ϕ is closed under composition and recursion. Hence each function in the list $f_1, \ldots, f_n$ belongs to ϕ. Since $f_n = f$, f belongs to ϕ.

Corollary 3.4.

Every primitive recursive function is computable.

In Chapter 4 we shall show how to obtain a computable function that is not primitive recursive. Hence it will follow that the set of primitive recursive functions is a proper subset of the set of computable functions.

Some Primitive Recursive Functions

$f(x, y) = x + y$

- We have to show how to obtain f from the initial functions using composition and recursion.

Initial Functions

$s(x) = x + 1$, $n(x) = 0$, $u_i^n(x_1, \ldots, x_n) = x_i$, $(1 \leq i \leq n)$

- Step 1: Define f recursively:

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

- Step 2: Use initial functions :

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

  where $g(x_1, x_2, x_3) = s(u_2^3(x_1, x_2, x_3))$.
- So, $f(x, y) = x + y$ is a primitive recursive function.

Some Primitive Recursive Functions

$h(x, y) = x \times y$

- We have to show how to obtain h from the initial functions using composition and recursion.

Initial Functions

$s(x) = x + 1$, $n(x) = 0$, $u_i^n(x_1, \ldots, x_n) = x_i$, $(1 \leq i \leq n)$

- Step 1: Define h recursively:

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

- Step 2: Use initial functions :

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

  where $g(x_1, x_2, x_3) = f(u_2^3(x_1, x_2, x_3), u_3^3(x_1, x_2, x_3))$ and $f(x_1, x_2) = x_1 + x_2$.
- So, $h(x, y) = x \times y$ is a primitive recursive function.

- We have to show how to obtain h from the initial functions using composition and recursion.

Initial Functions

$s(x) = x + 1$, $n(x) = 0$, $u_i^n(x_1, \ldots, x_n) = x_i$, $(1 \leq i \leq n)$

- Step 1: Define h recursively:

$$\begin{aligned} h(0) &= 0! = 1 \\ h(x+1) &= (x+1)! = x! \times s(x) \end{aligned}$$

- Step 2: Use initial functions :

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

  where
  $g(x_1, x_2) = s(x_1) \times x_2 = s(u_1^2(x_1, x_2)) \times u_2^2(x_1, x_2))$.
- So, $h(x) = x!$ is a primitive recursive function.

- We have to show how to obtain h from the initial functions using composition and recursion.

Initial Functions

$s(x) = x + 1$, $n(x) = 0$, $u_i^n(x_1, \ldots, x_n) = x_i$, $(1 \leq i \leq n)$

- Step 1: Define h recursively:

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

- Step 2: Use initial functions:

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

where $g(x_1, x_2, x_3) = u_2^3(x_1, x_2, x_3) \times u_1^3(x_1, x_2, x_3))$.

- So, $h(x, y) = x^y$ is a primitive recursive function.

Some Primitive Recursive Functions

Predecessor function

- $p(x) = \begin{cases} x - 1 & \text{if } x \neq 0 \\ 0 & \text{if } x = 0 \end{cases}$
- We have to show how to obtain p from the initial functions using composition and recursion.

Initial Functions

$s(x) = x + 1$, $n(x) = 0$, $u_i^n(x_1, \ldots, x_n) = x_i$, $(1 \leq i \leq n)$

- Step 1: Define p recursively:

$$\begin{aligned} p(0) &= 0 \\ p(t+1) &= t \end{aligned}$$

- So, $p(x)$ is a primitive recursive function.

Some Primitive Recursive Functions

$h(x,y) = x \dot{-} y$

- $h(x,y) = x \dot{-} y = \begin{cases} x - y & \text{if } x \geq y \\ 0 & \text{if } x < y \end{cases}$
- We have to show how to obtain h from the initial functions using composition and recursion.

Initial Functions

$s(x) = x + 1$, $n(x) = 0$, $u_i^n(x_1, \ldots, x_n) = x_i$, $(1 \leq i \leq n)$

- Step 1: Define h recursively:

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

- So, $h(x,y) = x \dot{-} y$ is a primitive recursive function.

Some Primitive Recursive Functions

- $h(x,y) = |x - y|$
  - $h(x,y) = |x - y| = x \dot{-} y + y \dot{-} x$
  - So, $h(x,y) = |x - y|$ is a primitive recursive function.
- $\alpha(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{if } x \neq 0 \end{cases}$
  - $\alpha(x) = 1 \dot{-} x$
  - or, $\alpha(0) = 1, \alpha(t+1) = 0$.
  - So, $\alpha(x)$ is a primitive recursive function.

- Predicates= Boolean-valued functions
- $x = y$ or $d(x,y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases}$
  - $d(x,y) = \alpha(|x - y|) \Rightarrow$ primitive recursive.
- $x \leq y \sim \alpha(x \dot{-} y) \Rightarrow$ primitive recursive.

Theorem 5.1. If P, Q are predicates that belong to a PRC class ϕ, then so are $\sim P$, $P \vee Q$, and $P \wedge Q$.

Proof.

- $\sim P = \alpha(P)$.
- $P \wedge Q = P \times Q$.
- $P \vee Q = \sim(\sim P \wedge \sim Q)$.

Corollaries:

- If P, Q are PR predicates, then so are $\sim P$, $P \vee Q$, and $P \wedge Q$.
- If P, Q are computable predicates, then so are $\sim P, P \vee Q$, and $P \wedge Q$.

Primitive Recursive Predicates

- Predicates= Boolean-valued functions
- $x = y$ or $d(x,y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases}$
    - $d(x,y) = \alpha(|x - y|) \Rightarrow$ primitive recursive.
- $x \leq y \sim \alpha(x \dot{-} y) \Rightarrow$ primitive recursive.

Theorem 5.1. If P, Q are predicates that belong to a PRC class ϕ, then so are $\sim P$, $P \vee Q$, and $P \wedge Q$.

Proof.

- $\sim P = \alpha(P)$.
- $P \wedge Q = P \times Q$.
- $P \vee Q = \sim(\sim P \wedge \sim Q)$.

- $x < y \equiv (x \leq y \wedge \sim(x = y)) \equiv \sim(y \leq x) \Rightarrow$ primitive recursive.

Theorem 5.4. (Definition by Cases).
If the functions g, h and the predicate P belong to a PRC class ϕ, then

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

belongs to ϕ.

Proof. $f(x_1, \dots, x_n) = g(x_1, \dots, x_n) \times P(x_1, \dots, x_n) + h(x_1, \dots, x_n) \times \alpha(P(x_1, \dots, x_n))$.

Primitive Recursive Predicates

Corollary 5.5.
If the functions $g_1, \ldots, g_m, h$ and the predicate $P_1, \ldots, P_m$ belong to a PRC class ϕ and $\forall 1 \leq i < j \leq m$ and $\forall x_1, \ldots, x_n$, $P_i(x_1, \ldots, x_n) \wedge P_j(x_1, \ldots, x_n) = 0$ then

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

belongs to ϕ.

Proof. (By induction on m)
Base step: $m = 1$ (Previous Theorem).
Induction hypothesis: It is true for m.

Proof. (Cont.)

Let

$$f(x_1,\ldots,x_n)=\begin{cases} g_1(x_1,\ldots,x_n) & \text{if } P_1(x_1,\ldots,x_n)\\ \vdots & \vdots \\ g_{m+1}(x_1,\ldots,x_n) & \text{if } P_{m+1}(x_1,\ldots,x_n)\\ h(x_1,\ldots,x_n) & \text{otherwise}\end{cases}$$

Let

$$h'(x_1,\ldots,x_n)=\begin{cases} g_{m+1}(x_1,\ldots,x_n) & \text{if } P_{m+1}(x_1,\ldots,x_n)\\ h(x_1,\ldots,x_n) & \text{otherwise}\end{cases}$$

Then

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

Done!

Iterated Operations and Bounded Quantifiers

Theorem 6.1. If $f(t, x_1, \ldots, x_n)$ belongs to a PRC class, then so do the functions

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

and

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

Proof. Note: we cannot use induction for the proof, because it proves that $\forall i, g(i, x_1, \ldots, x_n)$ belongs yo the PRC class.
Consider the following recursion:

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

Minimalization

Pairing Functions and Gödel Numbers