

Homework Questions

Question 2: Let \mathcal{P} be the following program:

```

IF X ≠ 0 GOTO A
Z ← Z + 1
IF Z ≠ 0 GOTO B
[A] X ← X - 1
   Y ← Y + 1
   IF X ≠ 0 GOTO A
[B] Y ← Y + 1
   Y ← Y + 1
    
```

What is the function $f(x)$ computed by \mathcal{P} ?

Answer: $f(x) = x+2$

September 22, 2009 Theory of Computation Lecture 5: Primitive Recursive Functions I 1

Homework Questions

Question 3: Write a program that computes $f(x) = 2x$ without using any macros. After the program terminates, the variable X must contain its initial value (the input).

Answer:

```

IF X≠0 GOTO A
Z ← Z+1
IF Z≠0 GOTO E
[A] X ← X-1
   Y ← Y+1
   Z ← Z+1
   IF X≠0 GOTO A
[B] Z ← Z-1
   Y ← Y+1
   X ← X+1
   IF Z≠0 GOTO B
    
```

September 22, 2009 Theory of Computation Lecture 5: Primitive Recursive Functions I 2

Recursion

Let k be some fixed number and

$$h(0) = k$$

$$h(t + 1) = g(t, h(t)) ,$$

where g is some given total function of two variables. Then we say that h is obtained from g by **primitive recursion**, or simply **recursion**.

Theorem 2.1: Let h be obtained as shown above, and let g be computable. Then h is also computable.

September 22, 2009 Theory of Computation Lecture 5: Primitive Recursive Functions I 3

Recursion

Proof: Obviously, the function $f(x) = k$ is computable. The program computing $f(x)$ simply consists of k times the instruction $Y \leftarrow Y+1$. This gives us the macro $Y \leftarrow k$. Now we can write a program that computes $h(x)$:

```

Y ← k
[A] IF X=0 GOTO E
   Y ← g(Z, Y)
   Z ← Z+1
   X ← X-1
   GOTO A
    
```

■

September 22, 2009 Theory of Computation Lecture 5: Primitive Recursive Functions I 4

Recursion

Here is a similar, but more complicated type of recursion:

$$h(x_1, \dots, x_n, 0) = f(x_1, \dots, x_n)$$

$$h(x_1, \dots, x_n, t + 1) = g(t, h(x_1, \dots, x_n, t), x_1, \dots, x_n) .$$

Here we say that that the function h of $n + 1$ variables is obtained by **primitive recursion** (or **recursion**) from the functions f (of n variables) and g (of $n + 2$ variables).

Theorem 2.2: Let h be obtained from f and g as shown above, and let f and g be computable. Then h is also computable.

September 22, 2009 Theory of Computation Lecture 5: Primitive Recursive Functions I 5

Recursion

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

```

Y ← f(X_1, \dots, X_n)
[A] IF X_{n+1}=0 GOTO E
   Y ← g(Z, Y, X_1, \dots, X_n)
   Z ← Z+1
   X_{n+1} ← X_{n+1}-1
   GOTO A
    
```

■

September 22, 2009 Theory of Computation Lecture 5: Primitive Recursive Functions I 6

PRC Classes

Now that we have learned about **composition** and **recursion**, let us consider the functions that can be constructed with these operations.

Let us define the following **initial functions**:

$$s(x) = x + 1$$

$$n(x) = 0$$

$$u_i^n(x_1, \dots, x_n) = x_i, \quad 1 \leq i \leq n.$$

The functions u_i^n are called **projection functions**.

For example, $u_2^3(x_1, x_2, x_3) = x_2$.

September 22, 2009

Theory of Computation
Lecture 5: Primitive Recursive Functions I

7

PRC Classes

Definition: A class of total functions \mathcal{C} is called a **PRC (primitive recursively closed) class** if

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

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

September 22, 2009

Theory of Computation
Lecture 5: Primitive Recursive Functions I

8

PRC Classes

Proof: We already know through Theorems 1.1, 2.1, and 2.2 that applying composition or recursion to computable functions results in further computable functions.

Therefore, we only have to show that the **initial functions** are computable.

Obviously, $s(x) = x + 1$ is computed by

$$Y \leftarrow X$$

$$Y \leftarrow Y + 1,$$

$n(x)$ is computed by the empty program, and

$u_i^n(x_1, \dots, x_n)$ is computed by

$$Y \leftarrow X_i$$

September 22, 2009

Theory of Computation
Lecture 5: Primitive Recursive Functions I

9

PRC Classes

Definition: 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.

Obviously, it follows that:

Corollary 3.2: The class of primitive recursive functions is a PRC class.

Furthermore, we have:

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

September 22, 2009

Theory of Computation
Lecture 5: Primitive Recursive Functions I

10

PRC Classes

Proof:

Part I: If a function belongs to every PRC class, then, by Corollary 3.2, it belongs to the class of primitive recursive functions.

Part II: Let f be a primitive recursive function and let \mathcal{C} be some PRC class. We want to show that f belongs to \mathcal{C} .

Since f is a primitive recursive function, there is a list f_1, f_2, \dots, f_n of functions such that $f_n = f$ and each f_i in the list is either

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

September 22, 2009

Theory of Computation Lecture 5: Primitive
Recursive Functions I

11

PRC Classes

Obviously, the initial functions belong to the PRC class \mathcal{C} .

Applying composition or recursion to functions in \mathcal{C} results in another function belonging to \mathcal{C} .

Thus each function in the list f_1, \dots, f_n belongs to \mathcal{C} .

Since $f_n = f$, f belongs to \mathcal{C} . ■

Corollary 3.4: Every primitive recursive function is computable.

Proof: By the theorem just proved, every primitive recursive function belongs to the PRC class of computable functions.

September 22, 2009

Theory of Computation Lecture 5: Primitive
Recursive Functions I

12

Another Word on PRC Classes

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

- the initial functions n , s , and u belong to C and
- a function obtained from functions belonging to C by recursion or composition also belongs to C .

Notice that this definition does **not** demand all functions in C to be obtained from n , s , and u by recursion or composition.

There could be **other functions** in C , say p and q , that cannot be obtained from n , s , and u . According to the definition, C is then still a PRC class if all functions obtained from n , s , u , p , and q by recursion or composition are also in C .

September 22, 2009

Theory of Computation Lecture 5: Primitive
Recursive Functions I

13

Another Word on PRC Classes

Now look at the definition of 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.

So here **no additional functions** such as p and q are allowed – all primitive recursive functions can be obtained from n , s , and u .

Therefore, the class of primitive recursive functions is the **minimal** PRC class. All primitive recursive functions are contained in every PRC class. However, PRC classes can contain additional functions (see previous slide).

September 22, 2009

Theory of Computation Lecture 5: Primitive
Recursive Functions I

14

Next Homework Questions

Question 1:

Write a program \mathcal{P} that computes $\Psi_{\mathcal{P}}^{(1)}(r) = r-2$ using no macros.

Notice that we only consider natural numbers! This means that, for example, the expression $1-2$ is undefined. Any program computing such an expression **must never halt**.

Question 2:

a) Write a program \mathcal{S} that computes $\Psi_{\mathcal{S}}^{(1)}(r) = r-4$. Use a macro based on program \mathcal{P} of the form $W \leftarrow p(V)$ (which has the effect $W \leftarrow V-2$).

b) Now comes the best part: Expand all macros in \mathcal{S} using the method we discussed in the lecture. You do not have to expand macros such as $V \leftarrow 0$ that this method uses.

September 22, 2009

Theory of Computation Lecture 5: Primitive
Recursive Functions I

15