CS 4510

Primitive Recursive Functions

Name: Abrahim Ladha

1.1 Primitive recursion

Long ago, we thought the chemical elements to be distinct, but we now know them as just permutations of the same fundamental building blocks; atoms. In the same way, you may notice that all functions which occur to you naturally are made up of smaller, more atomic functions, combined via composition. Take a moment to think of a set of small atomic functions which could not be made up of smaller functions.

These three functions we define to be primitive recursive:

$$\text{Zero: } N(x) = 0$$

$$\text{Successor: } S(x) = x + 1$$

$$\text{Projection: } U_i^n(x_1, ..., x_n) = x_i$$

A function is primitive recursive if it is obtained by applying two operations to other primitive recursive functions. Those two operations:

- Composition: If $g_1, ..., g_n, f \in PR$, then $f(g_1(\cdot), ..., g_n(\cdot)) \in PR$.
- Primitive Recursion: For $f, g \in PR$, we say h is PR when $h(0, x_1, ..., x_k) = f(x_1, ..., x_k)$ and $h(S(y), x_1, ..., x_k) = g(y, h(y, x_1, ..., x_k), x_1, ..., x_k)$.

Definition 1.1 *The set of primitive recursive functions (PR) is the smallest set containing N, S, U, and maintains closure under composition and primitive recursion.*

The composition operation is quite natural. For the primitive recursion operation, the first parameter of h is the depth of recursion. The first statement says that the base case is PR. The second statement is recursive. This is saying that $h(y, ...)$ being PR will imply that $h(S(y), ...)$ is PR.

To prove a function $h(y, x)$ is primitive recursive is similar to induction, and requires two steps. First show $h(0, x)$ is primitive recursive. Then assuming $h(y, x)$ is primitive recursive, show $h(S(y), x)$ is also primitive recursive. This is not so intuitive, and I believe is better learned by example:

Here we show that addition is primitive recursive.

- $add(0, x) = U_1^1(x)$
- $add(S(y), x) = S(add(y, x))$

Once you prove a function is primitive recursive, then you may use it to build up other more complex functions, and show those are primitive recursive. Here we will show multiplication, built from addition.

- $mul(0,x) = N(x)$
- $mul(S(y),x) = add(mul(y,x),x)$

1.2 PR $\subsetneq$ R

Every primitive recursive function is computable[1] but the converse is false. Not every computable function is primitive recursive. One example is the Ackermann function, constructed exactly to be not so:

$$A(m,n) = \begin{cases} n+1 & m = 0 \\ A(m-1,1) & m > 0, n = 0 \\ A(m-1, A(m,(n-1))) & m,n > 0 \end{cases}$$

It grows really fast. $A(4,3) = 2^{2^{2^{2^{2^{2}}}}} - 3$. This number does not fit into any calculator I have tried. The primitive recursive functions correspond to turing machines whose time complexity is bounded in the size of the input. If you think of code, any for loop must have the number of iterations fixed before the loop starts.

I don't have any particular feelings about the Ackermann function[2]. Here is another kind of proof of existence of a computable function which is not primitive recursive. We proceed by diagonalization. Let $\psi_1, \psi_2, ...$ be an ordering[3] of primitive recursive functions. Then let $\phi(x,y) = \psi_x(y)$. That is ϕ takes on two integral arguments, and is equal to the x'th primitive recursive function of variable y. Let $\psi(x) = \phi(x,x) + 1$. Assume to the contrary $\psi(x)$ is primitive recursive. Then there is index for it. There exists n such that $\psi(x) = \psi_n(x)$. Then by definition, $\psi(x) = \phi(n,x)$. Take $x = n$. It then follows that

$$\phi(n,n) = \psi_n(n) = \psi(n) = \phi(n,n) + 1 \tag{1.1}$$

A contradiction! Therefore, ψ is not primitive recursive.

[1]Exercise, see 9.

[2]You may find a clean proof that $A \notin PR$ here: http://www.cs.tau.ac.il/ nachumd/term/42019.pdf

[3]Convince yourself that PR is a countable set, so there must exist some bijection $\mathbb{N} \to PR$. In the next sheets we will learn about Gödel numberings. This ordering is a weaker version of that.

1.3 Problems

Turn in 6 of the first 8 problems along with 9. You may use the results of problems in your solutions to other problems.

1. Let $P(0) = P(1) = 0$ and $P(S(x)) = x$ otherwise. Prove that P is primitive recursive.
2. Prove that the identity function, $id(x) = x$ is primitive recursive.
3. Prove that for any constant c, $f(x) = c$ is primitive recursive
4. Prove $x!$ is primitive recursive
   - $0! =$
   - $S(x)! =$
5. Prove x^y is primitive recursive
   - $exp(0, x) =$
   - $exp(S(y), x) =$
6. Let $\multimap$ represent truncated subtraction. If $x - y$ is non-negative, then $x \multimap y = x - y$, and $x \multimap y = 0$ if $x < y$. Prove truncated subtraction is primtive recursive.
7. Prove that $\max(x, y)$ is primitive recursive.
8. Compute[4] A(1,2) and A(3,3).
9. Prove every primitive recursive function is computable. (Hint: use induction on the depth of recursion. The base case is one of N, S, U)
10. Let the Collatz function be

$$C(n) = \begin{cases} n/2 & \text{if even} \\ 3n + 1 & \text{if odd} \end{cases} \tag{1.2}$$

Let $D(n)$ be the program such that $x = n$, while $x \neq 1, x \leftarrow C(x)$. Prove that if D is primitive recursive, then the Collatz conjecture is true.

Further Reading

[1] Martin Davis. *Computability and Unsolvability. 1982 ed.* 1958.

[4]It is okay to use programming, just mention how you did it.