The primitive recursive functions

Raúl Rojas

January 20, 1998

1 Introduction

2 Basis functions and operations

The primitive recursive functions are defined by starting with the following small set of basis functions which map natural numbers to natural numbers:

- The zero function $z : x \mapsto 0$,
- The succesor function s
- The projection (or identity) functions:

  $$p_i^j(x_1, x_2, \ldots, x_j) = x_i$$

  Since the superindex j is implicitly given by the number of arguments of the function, we simplify the notation by writing only the subindex of the projection function. It should nevertheless be clear that p_1 with one argument and p_1 with two arguments represent two different projection functions, namely p_1^1 and p_1^2.

It is sometimes useful to think in terms of what these functions represent in a modern computer: the zero function clears the accumulator of the machine; given a number, the succesor function gives us the next number; the projection functions are similar to the LOAD instructions that allow us to select a specific number from a memory array.

There are only two constructive rules that allow us to generate new primitive recursion functions:

- *Composition rule:* If g and $f_1, \ldots, f_k$ are previously defined primitive recursive functions and f has the form

  $$f(x_1, x_2, \ldots, x_n) = g(f_1(x_1, x_2, \ldots, x_n), \ldots, f_k(x_1, x_2, \ldots, x_n))$$

  then f is primitive recursive.

- *Primitive recursion:* If ψ and χ are previously defined primituve recursive functions and ϕ has the form

$$\begin{aligned} \phi(0, x_1, x_2, \ldots, x_n) &= \psi(x_1, x_2, \ldots, x_n) \\ \phi(s(y), x_1, x_2, \ldots, x_n) &= \chi(\phi(y, x_1, x_2, \ldots, x_n), y, x_1, x_2, \ldots, x_n) \end{aligned}$$

  then ϕ is primitive recursive. The n arguments $x_1, \ldots, x_n$ are optional, that is, n can be zero. In this case ψ can be a natural number.

The definition of other functions can be simplified by taking note of of the following implicit properties of the composition rules:

- Arguments can be rearranged in any desired order. For example: if g is primitive recursive, then

$$f(x_1, \ldots, x_n) = g(x_{i_1}, \ldots, x_{i_n})$$

  is primitive recursive, where $i_1, \ldots, i_n$ is any permutation of the indices $1, 2, \ldots, n$. This holds because f can be written as

$$f(x_1, \ldots, x_n) = g(p_{i_1}(x_1, \ldots, x_n), \ldots, p_{i_n}(x_1, \ldots, x_n))$$

  which complies with the general form of the composition rule.

- Arbitrary nesting of primitive recursive functions produces primitive recursive functions.

3 Some useful functions

Functions of a single argument

$$\begin{aligned} \phi(0) &= k \\ \phi(s(y)) &= \chi(\phi(y), y) \end{aligned}$$

The predecessor function v can be defined using $\chi = p_2^2$. The zero function can be defined using $\chi = p_1^2$. The boolean function t_0 which tests equality with zero can be defined using $\chi(a, b) = clr(p_2^2(a, b))$

Addition

$$\begin{aligned} \oplus(0, b) &= p_1^1(b) \\ \oplus(s(a), b) &= s(p_1^3(\oplus(a, b), a, b)) \end{aligned}$$

Multiplication

$$\begin{aligned} \otimes(0,b) &= z(b) \\ \otimes(s(a),b) &= \oplus(p_1^3(\otimes(a,b),a,b),p_3^3(\otimes(a,b),a,b) \end{aligned}$$

If-then-else

$$\begin{aligned} I(0,b,c) &= p_2^2(b,c) \\ I(s(a),b,c) &= p_3^4(I(a,b,c),a,b,c) \end{aligned}$$

Division

$$\oslash(a,b) = \div(a,a,b)$$

$$\begin{aligned} \div(0,a,b) &= z(p_1^2(a,b)) \\ \div(s(q),a,b) &= I(qb <= a,q,\div(q,a,b)) \end{aligned}$$

Wurzel

$$\begin{aligned} sqrt(0,b) &= 0 \\ sqrt(s(a),b) &= I(a^2 <= b,a,sqrt(a,b)) \end{aligned}$$

4 Lists

The packing function:

$$\alpha(m,n) = \frac{(m+n)(m+n+1)}{2} + m$$

The unpacking functions:

$$\theta(z) = \frac{\sqrt{8z+1}-1}{2}$$

$$\mu(z) = z - \frac{\theta(z)(\theta(z)+1)}{2}$$

$$\eta(z) = \theta(z) - \mu(z)$$

References

[1] Minsky

[2] R. Péter, *Recursive Functions*, Academic Press, New York, 1967.

[3] R. Gandy, "The confluence of ideas in 1936", in: R. Herken, *The Universal Turing Machine – A Half Century Survey*, Oxford University Press, Oxford, 1988.

[4] J. B. Rosser, "Highlights of the history of the lambda calculus", *Annals of the History of Computing*, Vol. 6, N. 4, pp. 337-349.