LOS ALGORITMOS COMO INSTRUMENTO DE LA MATEMATICA[1]

Ing. Luis Roberto Ojeda Ch.[2]

Resumen

El concepto de algoritmo es fundamental en múltiples campos de la actividad humana. Se relacionan en este artículo los conceptos de función y de algoritmo para luego caracterizar un tipo especial de función: el de las recursivas. Finalmente se establece la Tesis de Alonzo Church que indica que sólo para las funciones recursivas pueden obtenerse algoritmos de evaluación.

Qué es una función

$w = f(x_1,x_2,....,x_n)$

Es una forma de denotar que:

a) w es una variable dependiente.

b) $x_1,x_2,....,x_n$ son argumentos o variables independientes.

w es una *función* si se conoce la ley que para diversos conjuntos de valores concretos:

$\{x_1,x_2,....,x_n\}$

prefija valores determinados de la variable w.

Para que w sea una función es preciso que la ley en cada caso defina un único valor.

En cuanto a la ley aplicable para definir w como función, las matemáticas no imponen restricción alguna.

Como ley puede servir cualquier *algoritmo*, y en tal caso la función se denomina *calculable*.

Así, para la función

$w = f(x_1,x_2,....,x_n)$

Tendremos el esquema:

$$\text{Datos} \left[\begin{array}{l}\text{esto es, argu-}\\ \text{mentos actuales}\end{array}\right] + \text{Algoritmo} = \text{Resultado}$$

Resultado puede caracterizarse como un agregado de componentes, y lo importante es que se produce un único agregado.

Es adecuado recordar que cuando w en

$w = f(x_1,x_2,....,x_n)$

es una función, se distinguen tres partes:

a) Nombre de la función: w.

b) Escritura funcional: $w = f(x_1,x_2,....,x_n)$

c) Valor (uno solo).

1 Por extensión: Método o manera de pensar. Enciclopedia Larousse.

2 M.S. Computer Science, U.C.L.A.

La letra f se llama símbolo funcional, y aparte de f pueden emplearse otros signos como símbolo funcional.

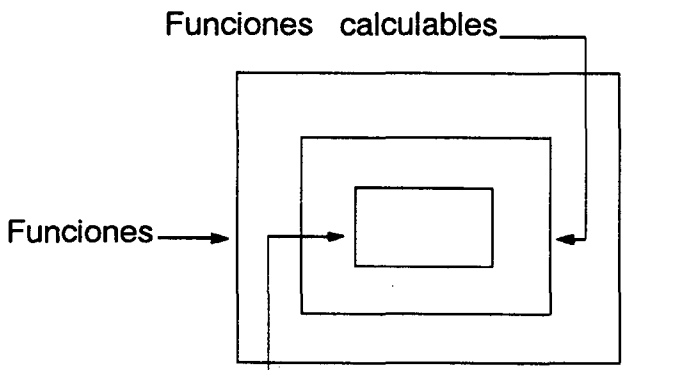
Hacemos la aclaración de que los argumentos estarán siempre codificados como números naturales en el presente artículo.

Las funciones calculables y las funciones recursivas

Dentro de las funciones calculables hay una clase muy importante que es la de las funciones recursivas.

Para las funciones recursivas siempre existen algoritmos que son leyes de su planteamiento. Ellos reciben el nombre de *algoritmos que acompañan a las funciones recursivas*.

Un aspecto muy importante de tales algoritmos es que operativamente ellos constituyen la forma de evaluación de las funciones.



funciones recursivas, con algoritmos que:

a) son leyes de su planteamiento.

b) constituyen su forma de evaluación.

Las leyes de planteamiento para toda función recursiva, ésto es, los algoritmos acompañantes

Se reconoce un subconjunto finito elemental de funciones recursivas básicas.

Sus símbolos funcionales son: φ, ψ, λ

Nuestro conjunto de recursivas básicas tiene entonces sólo tres componentes:

recursivas básicas = { φ_n , $\psi_{n.i}$, λ}

Cada uno de los tres componentes es un tipo.

Tipo 1: φ_n

Representa funciones de cualquier cantidad de variables independientes.

Su valor es siempre 0.

Se designan con φ_n donde n es la cantidad de variables independientes o argumentos.

Ejemplo:

$g(x) = 0$
$w(x,y,z) = 0$
$z(x_1,x_2,...,x_n) = 0$

Dentro del tipo, $g(x) = 0$ es:

$\varphi_1(x)$
$w(x,y,z) = 0$ es:
$\varphi_3(x,y,z)$

El algoritmo acompañante para las funciones Tipo 1 (que es además operativamente su forma de evaluación), es:

> **Si el símbolo funcional tiene la forma φ_n, a cualquier conjunto de valores argumento se le pone en correspondencia el valor 0.**

Así en el ejemplo anterior:

$\varphi_1(0) = 0$

$\varphi_1(26) = 0$
$\varphi_3(3,6,9) = 0$
$\varphi_n(5,9,16,...,837) = 0$

Si n es nulo se tendrá:

$\varphi_0( ) = 0$

o brevemente, $\varphi_0 = 0$

Tipo 2: $\psi_{n,i}$

Su sentido es de escogencia entre los argumentos.

Se les denomina funciones idénticas de cualquier número de variables independientes.

Su forma es:

$\psi_{n,i}$

donde:

n es la cantidad de argumentos, $n > 0$.

i designa uno de los argumentos $1 \leq i \leq n$

Algoritmo acompañante:

> Si el símbolo funcional es de la forma $\psi_{n,i}$, el valor de la función es el de i-ésima variable independiente, de izquierda a derecha.

Ejemplo:

$w = \psi_{3,2}(x,y,z)$
$V = \psi_{1,1}(n)$

Para w(5,8,0) el valor es 8.
Para v(6) el valor es 6.

Tipo 3: λ

Su sentido se asocia con un seguimiento, ésto es, con la obtención de un siguiente.

Hace necesario que los integrantes del dominio y del codominio tengan naturaleza común, y que haya un mapa sobre los números naturales, que identifique al siguiente (y al actual), de cualquiera de ellos.

Algoritmo acompañante:

> Si el símbolo funcional es de la forma λ el valor de la función es el siguiente al argumento.

La operación de obtener un continuador la designaremos con ayuda de un asterisco:

$0^* = 1$
$1^* = 2$
$10^* = 11$
$\lambda(x) = x^*$
$\lambda(5) = 6$

Operadores

Para poder realizar cualquier función recursiva se requiere aparte del conjunto finito elemental de funciones recursivas básicas:

$\{ \varphi_n, 1_{n,i}, \lambda \}$

y de sus algoritmos acompañantes, de alguna forma clara, precisa y algorítmica de combinarlas.

Se reconocen tres procedimientos formales (algoritmos), que *a partir de funciones recursivas* (p.e. las básicas!), *obtienen nuevas funciones recursivas*.

Tales procedimientos son:

a) Operador de superposición o de sustitución, que denotaremos con S.

b) El operador recursivo, que denotaremos con R.

c) Operador de construcción respecto del primer cero, que denotaremos con μ.

El operador de superposición o sustitución

Utilizando el símbolo ::= que se lee "se define como" y denotando con S al operador, una función Φ puede construirse así:

$\Phi ::= S[F;f_1,f_2,...,f_n]$

Se supone que F es una función de n argumentos.

La sustitución S hace que los argumentos de F sean respectivamente $f_1,f_2,...,f_n$, entendiéndose que $f_1,f_2,...,f_n$ son funciones recursivas.

El algoritmo correspondiente a S es:

> Los valores de las funciones $f_1,f_2,...,f_n$ se toman como los valores de los argumentos de F y se calcula ésta. F debe ser una función recursiva.

Ejemplo:

F = $\lambda(x)$, recursiva
f1 = $\lambda(y)$, recursiva
$\Phi ::= S[\lambda(x); \lambda(y)]$

con lo cual

$\Phi = \lambda(\lambda(y))$
$\Phi = y^{**}$

en particular

$\Phi(5) = 7$

El operador recursivo R

La función construida con éste operador la denominaremos f.

Existirán dos funciones recursivas f_1 y f_2, sobre las que trabajará R.

$f ::= R[f_1,f_2;y,(z))$
f será una función en n variables,
$n >= 1$
f_1 es una función de n-1 variables,
f_2 es una función de n+1 variables.

Las variables involucradas en f, f_1 y f_2 tienen las siguientes características:

FUNCION	VARIABLES INVOLUCRADAS (INDEPENDIENTES)	NUMERO DE VARIABLES
f_1	$X_1, X_2, ..., X_{n-1}$	n-1
f	$X_1, X_2, ..., X_{n-1}, y$	n
f_2	$X_1, X_2, ..., X_{n-1}, y, z$	n+1

Esto es, las tres funciones tienen en común las variables independientes $x_1,x_2,...,x_n$.

La función f, así como la función f_2, tienen una variable extra, que es *y*.

La función f_2 tiene otra variable extra que es *z*.

y, la variable extra en f se llama principal.

z, la otra variable extra en f_2, se llama auxiliar.

El algoritmo correspondiente a R dice:

> El valor de f para y = 0 es igual al valor de f_1.
> El valor de f para cada valor sucesivo de y será el de f_2 con y = valor anterior de y, con z = f (valor anterior de y).

Ejemplo:

$pr(x) ::= R[\varphi_0,\psi_{2,1}(x,y);x,(y)]$

En este caso:

n = 1

variable principal es x (está en pr)

variable auxiliar es y (sólo está en f_2)

$pr(0) = \varphi_0 = 0$
$pr(i^*) = \psi_{2,1}(i,pr(i))$
pr(i*) = i, para cualquier i posterior a 0.

Obsérvese:

pr(0) = 0
pr(1) = 0
pr(2) = 1

etc.

pr es la función precedente:

precedente de 0 es 0
precedente de 1 es 0
precedente de 2 es 1, etc.

Téngase en cuenta además, que f_1 (esto es φ_0), y f_2 (o sea $\psi 2.1(x,y)$) son funciones recursivas, como se ha indicado en la definición de R.

Ejemplo:

$s(x,y) ::= R[\psi_{1,1}(x), \lambda(\psi_{3,3}(x,y,z));y , (z)]$
f_1 es $\psi_{1,1}(x)$ y n = 2;
la principal es y
la auxiliar es z.

Este ejemplo mostrará que el papel de la variable extra auxiliar es de apoyo para el cálculo de f (que es s(x,y)).

Aplicando el algoritmo pertinente, tenemos :

$s(x,0) = \psi_{1,1}(x) = x$

y en general,

$s(x,i^*) = (\psi_{3,3}(x,y,z))$

pero con: y = i
z = s(x,i)
Para $i^* = 1$
$s(x,i^*)$ = siguiente de s(x,0)
$s(x,i^*)$ = siguiente de x
s(x,1) = x + 1.

Así mismo:

s(x,2) = x + 2
....
s(x,y) = x + y

Esto es, en el mundo de los números naturales

f(x,y) = x + y

es una función recursiva.

Obsérvese que en el ejemplo f_1 es recursiva (es $\psi_{1,1}(x)$) y f_2 también lo es:

$f2 ::= s[\lambda(p);\psi_{3,3}(x,y,z)]$

Téngase en cuenta también que el mecanismo de R es tal que las n-1 variables propias de f_1 influyen en el cálculo del primer valor de f y que de allí en adelante, el valor de f está esencialmente caracterizado por las variables extras.

El operador de construcción con respecto del primer cero μ

Construye una nueva función recursiva de n argumentos a partir de una función recursiva de n + 1 argumentos.

El argumento que desaparece es el auxiliar y él es utilizado al ejecutar el operador que se denomina μ.

$f(x) ::= \mu[g(x,y)= 0;(y)]$

y es el argumento auxiliar.

g(x,y) es la función recursiva de n + 1 argumentos

f(x) es la función construida.

Para un argumento de x, p.e. a, la evaluación de f(x) se hace así:

Se toma g(a,y) y se hace y sucesivamente igual a 0,1,2,3,... hasta encontrar el primer valor y_0 para y tal que

$g(a,y_0) = 0$

Entonces

$f(a) = y_0$

Esto establece el algoritmo correspondiente al operador, y en lugar de x podríamos tener en general:

$f(x_1,x_2,...,x_n) ::= \mu[g(x_1,x_2,...,x_n,y)=0;(y)]$

Para algunos valores de a puede no existir valor de y tal que

$g(a,y) = 0$

En tal caso la secuencia de cómputo podría realizarse indefinidamente sin lograr f(a).

Ejemplo:

sean:

$g(x,y) = \psi_{2,1}(x,y)$
$f(x) = \mu[\psi_{2,1}(x,y);(y)]$
$f(0) = 0$

f(1) no existe, y en general, f(i) no existe para i 0, puesto que

$\psi_{2,1}(1,0) = 1$
$\psi_{2,1}(1,1) = 1$, etc.

Esta y toda otra función recursiva no determinada para todos los posibles valores de los argumentos se llama parcialmente recursiva. El siguiente es otro caso de función parcialmente recursiva.

Ejemplo:

$f(x) ::= \mu[residuo(x,2)+ y;(y)]$

Si x es par su residuo al dividirlo por 2 será 0, y por tanto el primer valor de y para el cual

$g(x,y)= residuo(x,2) + y$

se hace 0 es 0.

Si x es impar no existirá valor para y y f(x) será indefinido.

Conclusión

Son recursivas las funciones básicas y cualesquiera funciones que se obtienen de ellas mediante una cantidad finita de operaciones de sustitución, recursivo y de construcción respecto del primer cero.

Las funciones que se producen sin aplicar este último operador se denominan primitivamente recursivas, y son las más sencillas.

Muchas funciones conocidas son recursivas.

P.e. $y = x + 1$

es recursiva:

$y = (x)$

Según opinión del matemático norteamericano Alonzo Church el concepto de función recursiva agota el concepto de función calculable.

Esta opinión no refutada por la experiencia matemática actual constituye la tesis de Church[3].

BIBLIOGRAFIA

1. Machines, languages and computation. Denning, Dannis, Qualitz. Prentice Hall. 1978.
2. Algoritmos a nuestro alrededor. N. Krinitski, Mir. Moscú 1984.

3 Church, A. [1936] An unsolvable problem of elementary number theory am I. Math.