LA COMPUTADORA: UN LABORATORIA DE IDEAS PARA LOS MATEMÁTICOS

JOSÉ MANUEL GÓMEZ SOTO

UNIVERSIDAD LA SALLE

PROGRAMA DE DAVID HILBERT

CONGRESO INTERNACIONAL DE MATEMÁTICAS, PARIS 1900.

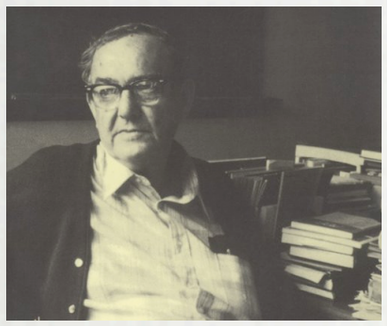
PROBLEMA 10: DADA UNA ECUACIÓN DIOFANTINA CON CUALQUIER NÚMERO DE INCÓGNITAS Y CON COEFICIENTES ENTEROS, DISEÑAR UN PROCEDIMIENTO CON EL CUAL SE PUEDA DETERMINAR, EN UN NÚMERO FINITO DE OPERACIONES, SI LA ECUACIÓN TIENE SOLUCIONES ENTERAS.

MÁQUINA DE TURING

ALAN TURING: ABORDO EL 10 PROBLEMA DE HILBERT EN TÉRMINOS DE UN DISPOSITIVO TEÓRICO LLAMADO LA ´MAQUINA DE TURING ´. ALGORÍTMICAMENTE ERA INDECIDIBLE SABER SI LA MÁQUINA SE IBA A DETENER O NO, AL RESOLVER DICHO PROBLEMA.

LA MÁQUINA DE TURING ES LA DEMOSTRACIÓN MÁTEMÁTICA DE LA POSIBILIDAD DE LAS COMPUTADORAS.

SISTEMAS SIMBÓLICOS FÍSICOS



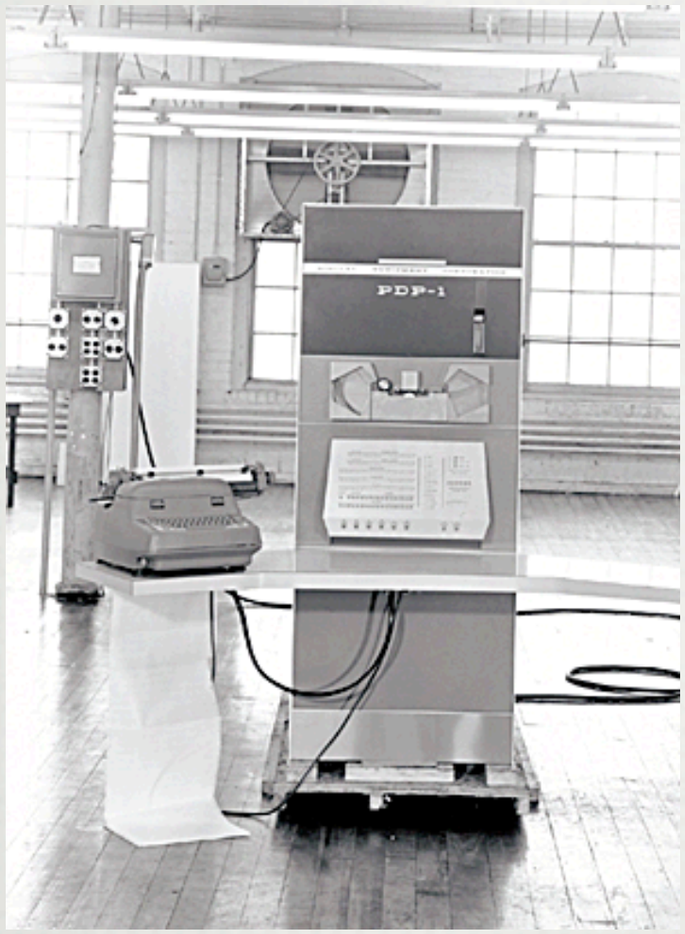
Hebert Simons

En algún nivel de la abstracción humana los procesos del pensamiento pueden ser modelados como relaciones entre símbolos.

Allen Newell

Newell A., Simon A. H. : Computer Science as empirical inquiry: symbols and search. Communications of ACM, Vol 19, Iss: 3, March 1976, pages 113-126.

LISP



J. McCarthy: Recursive Functions of Symbolic Expressions and their Computation by Machine. MIT AI Lab., AI Memo No. 8, Cambridge March 1959.

ENFOQUES

- [ ] PARA LOS MATEMÁTICOS ES NATURAL ESCRIBIR PROGRAMAS

- [ ] LA COMPUTADORA COMO UNA HERRAMIENTA GENERADORA DE IDEAS

- [ ] LA COMPUTADORA COMO UNA HERRAMIENTA PEDAGÓGICA: EN BÚSQUEDA DEL SIGNIFICADO DE LOS CONCEPTOS.

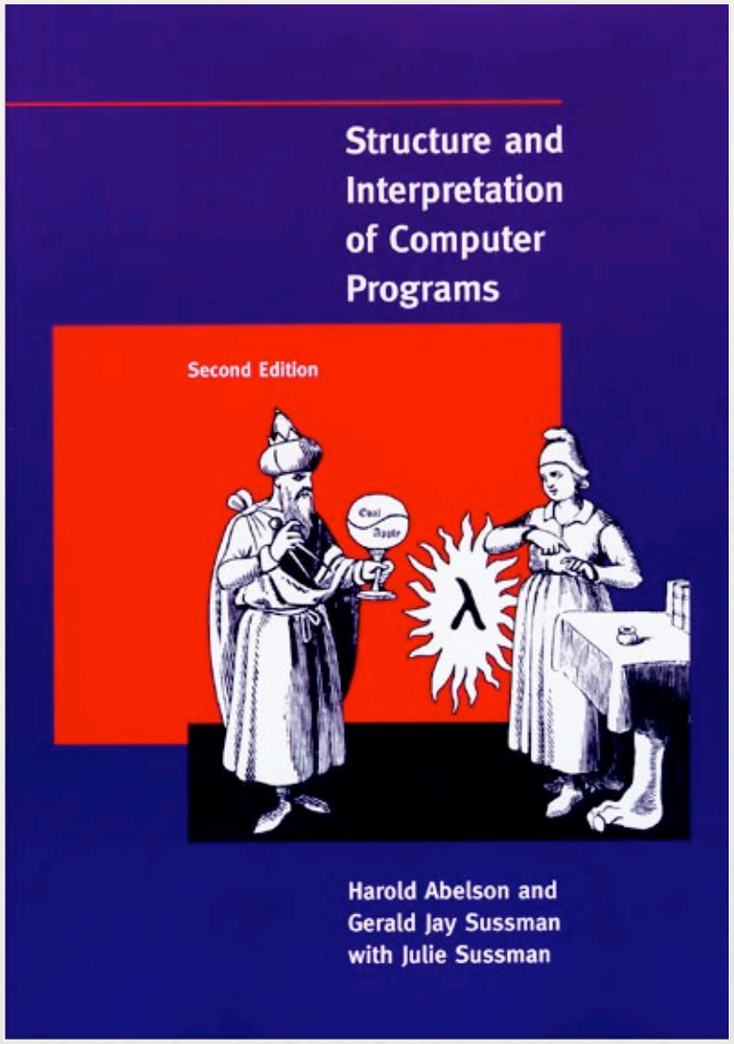
PROCESAMIENTO SIMBÓLICO

PROCESAMIENTO SIMBÓLICO

- ☐ AXIOM
- ☐ DERIVE
- ☐ MACSYMA
- ☐ MUMATH
- ☐ MATLAB
- ☐ REDUCE
- ☐ MATHEMATICA

Es una de las áreas fundamentales en el área de la computación que produce sistemas poderosos para el cómputo exacto y el rasonamiento formal con expresiones en forma simbólica.

SCHEME
DIALECTO DE LISP



PROGRAME AHORITA APRENDA DESPUÉS

OPERADORES-PRIMITIVOS

DATOS-PRIMITIVOS

EXPRESIÓN (+ 2 3)

COMPOSICIÓN

(+ 2 3 (-23) 4 1 (*2(/24)))

ABSTRACCIÓN

(DEFINE PI 3.141516)

(DEFINE AREA (+ 2 3 (-23) 4 1 (*2(/24))))

PROGRAME AHORITA APRENDA DESPUÉS

CREACIÓN DE PROCEDIMIENTOS

(lambda(x) (* x x))

ABSTRACCIÓN

(define cuadrado (lambda(x) (* x x)))

PROGRAME AHORITA APRENDA DESPUÉS

INSTRUCCIONES DE CONTROL

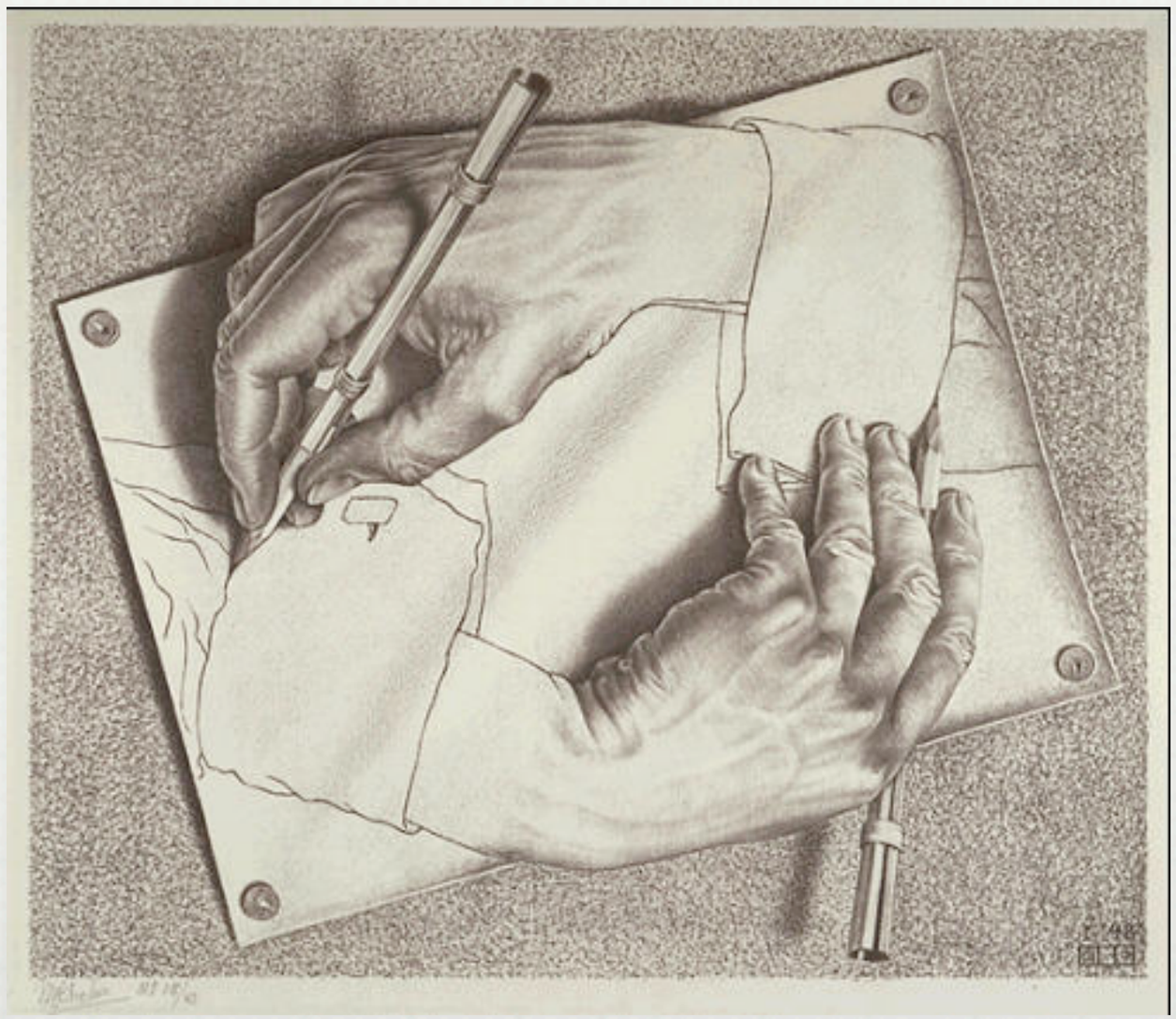
IF

(if exp-bool exp1 exp2)

COND

(cond (exp-bol1 exp1)
(exp-bol1 exp2)
:
(else expn))

RECURSIVIDAD



EXPONENCIAL

$$b^n$$

$$b^n = \begin{cases} 1 & \text{si } n = 0 \\ bb^{n-1} & \text{otro caso} \end{cases}$$

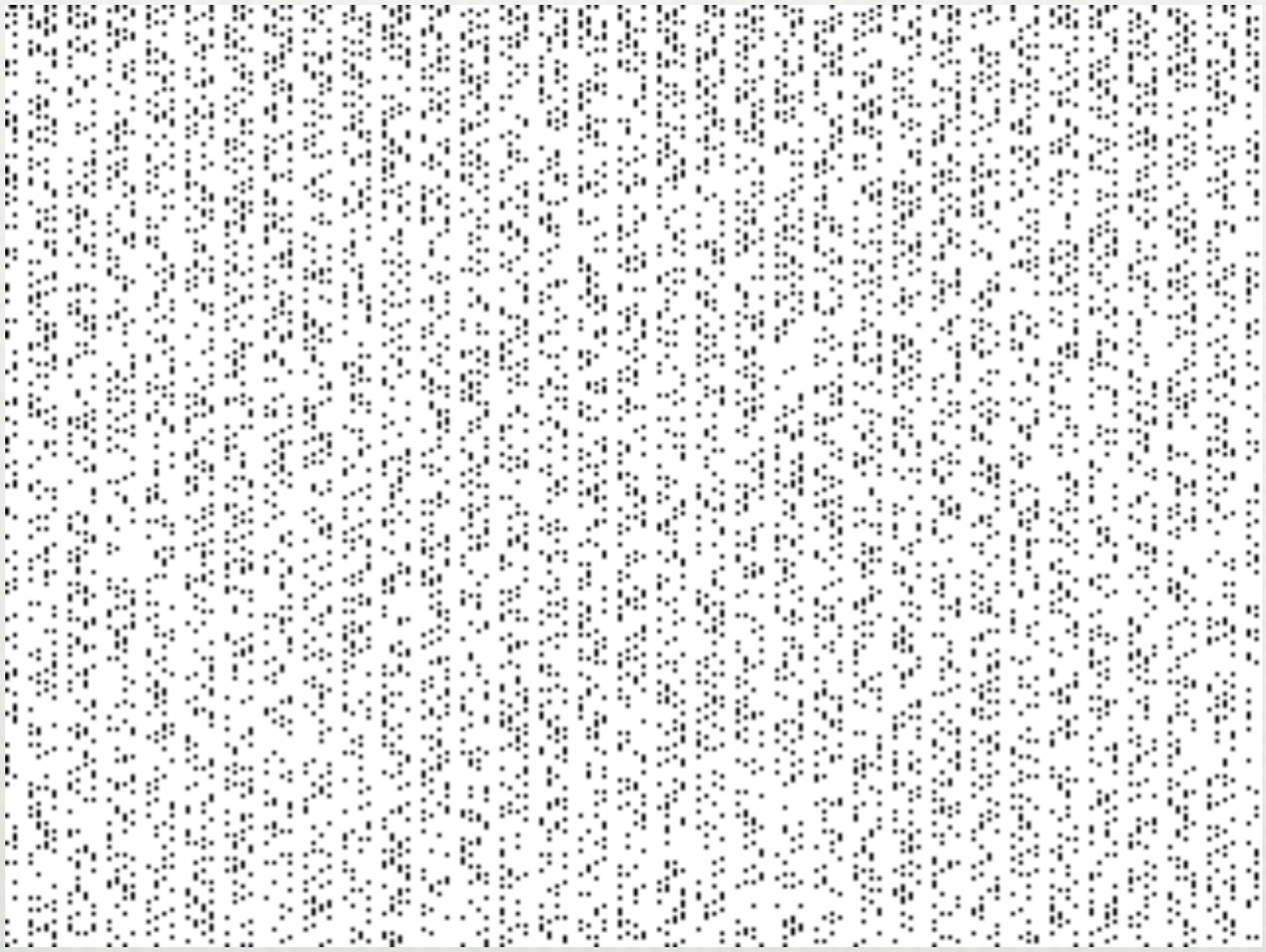
```
(define exp (lambda(b n)
(if (= n 0)  1 (* b (exp b (- n 1))))))
```

NÚMEROS PRIMOS

- ☐ PRIMEROS, FUNDAMENTALES, A PARTIR DE LOS CUALES SE FORMAN LOS OTROS NÚMEROS.
- ☐ NÚMEROS: COMPUESTOS Y PRIMOS.
- ☐ SON INFINITOS, NO MUESTRAN UN ORDEN.
- ☐ DISTRIBUCIÓN: $D(P) = N/LN(N)$
- ☐ MISTERIOS: ¿SON INFINITAS LAS PAREJAS DE INFINITOS?

NÚMEROS PRIMOS

Un número primo es aquel cuyo mínimo divisor después del uno es el mismo.



NÚMEROS PRIMOS

Un número primo es aquel cuyo mínimo divisor después del uno es el mismo.

```
(define primo? (lambda(n)
(= (minimo-divisor n) n)))
```

NÚMEROS PRIMOS

Mínimo divisor

```
(define minimo-divisor (lambda(n)
                         (busca-divisor n 2)))
```

```
(define busca-divisor (lambda(n posible-divisor)
(if (esdividido? n posible-divisor)
posible-divisor
(busca-divisor n (+ posible-divisor 1)))))
```

```
(define esdividido? (lambda(n posible-divisor)
(= (remainder n posible-divisor) 0)))
```

SERIES

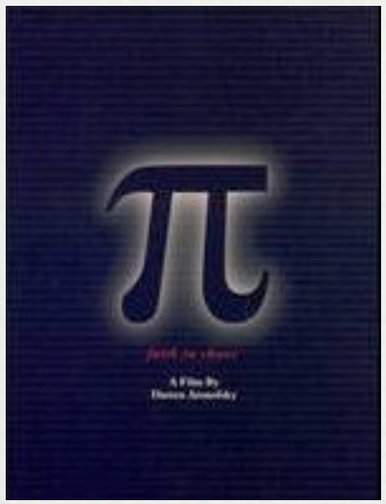
$$1 + 2 + 3 + 4 + 5 + 6 + 7 + \cdots + n$$

```
(define suma-enteros (lambda(a b)
(if (> a b)
0
(+ a (suma-enteros (+ a 1) b )))))
```

SERIES

$$1 + 4 + 9 + 16 + 25 + 36 + 49 + 64 \cdots + n^2$$

```
(define suma-enteros (lambda(a b)
                       (if (> a b)
                           0
                           (+ (cuadrado a) (suma-enteros (+ a 1) b )))))
```


A Film By

PI

$$\frac{\pi}{8} = \frac{1}{1*3} + \frac{1}{5*7} + \frac{1}{9*11} \cdots$$

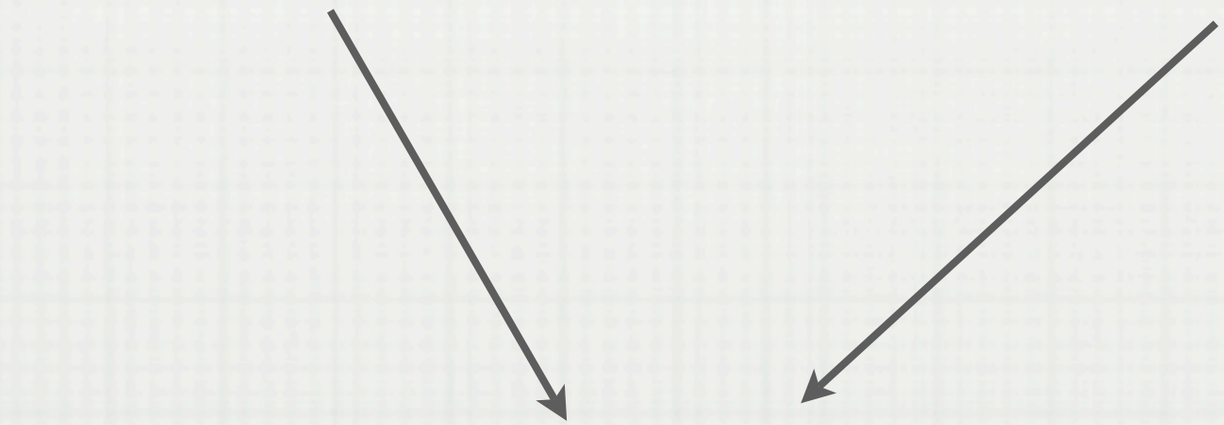
```
(define suma-pi (lambda(a b)
(if (> a b)
0
(+ (/ 1.0 (* a (+ a 2))) (suma-pi (+ a 4) b)))))

(define pi (* 8 (pi-suma 1 10000)))
```

SERIES

```
(define suma-enteros (lambda(a b)
  (if (> a b)
      0
      (+ a (suma-enteros (+ a 1) b )))))
```

```
(define suma-enteros (lambda(a b)
  (if (> a b)
      0
      (+ (cuadrado a) (suma-enteros (+ a 1) b )))))
```



```
(define suma (lambda(a b termino siguiente)
  (if (> a b)
      0
      (+ (termino a) (suma (siguiente a) b termino siguiente )))))
```

SERIES

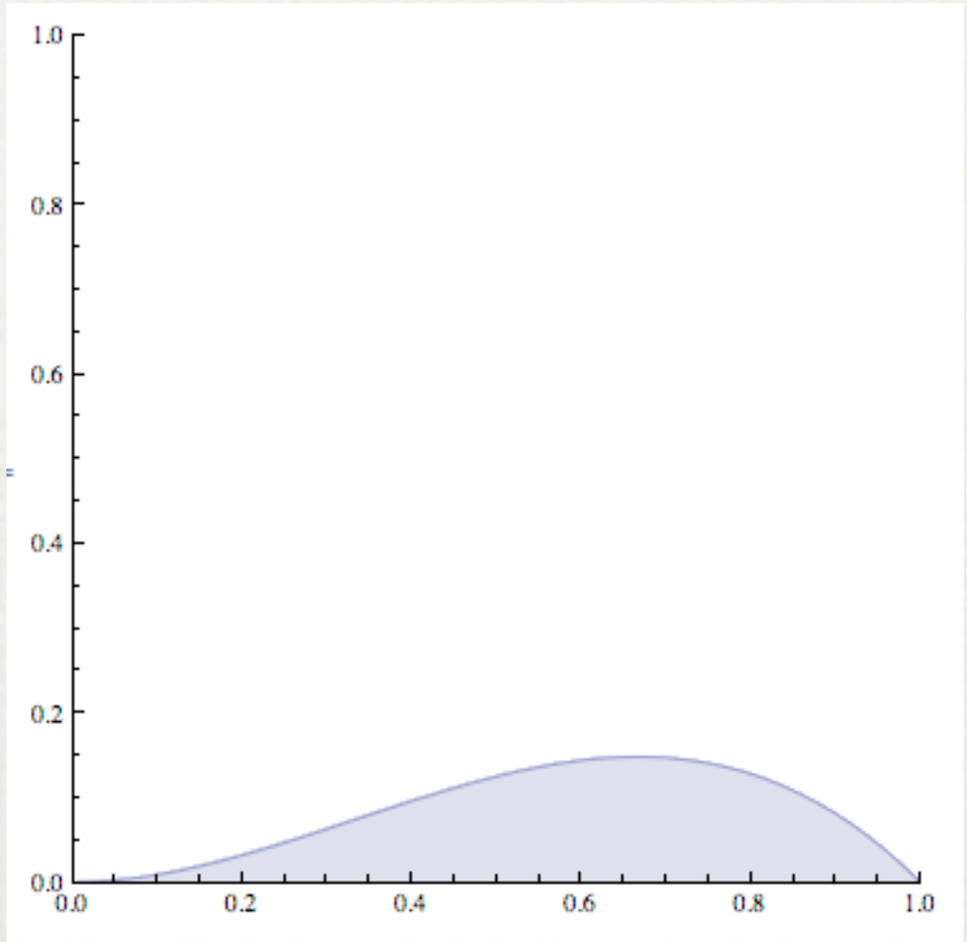
$$1+2+3+4+5+6+7+\cdots+n$$

```
(suma 1 100 (lambda(x) x) (lambda(x) (+ x 1)))
```

$$1+4+9+16+25+36+49+64\cdots+n^2$$

```
(suma 1 100 (lambda(x) (* x x)) (lambda(x) (+ x 1)))
```

INTEGRACIÓN NUMÉRICA



$$\int_0^1 (x^2 - x^3)dx = 0.0833333$$

INTEGRACION NUMÉRICA

$$\int_a^b f = [f(a+\frac{dx}{2})+f(a+\frac{dx}{2}+dx)+f(a+\frac{dx}{2}+2dx)+f(a+\frac{dx}{2}+3dx)+\ldots]dx$$

```
(define integral (lambda(a b f dx)
                   (* (suma (+ a (/ dx 2)) b f
                            (lambda(x) (+ x dx))) dx)))
```

```
(integral 0 1 (lambda(x) (- (expt x 2) (expt x 3))) 0.0001)
```

DERIVACIÓN NUMÉRICA

$$f'(x) = \lim_{dx \mapsto 0} \frac{f(x+dx) - f(x)}{dx}$$

```
(define derivada (lambda(f)
        (lambda(x) (/ (- (f (+ x dx)) (f x) dx))

(define dx 0.001)
```

DERIVACIÓN SIMBÓLICA

1) $$\frac{dc}{dx} = 0$$

2) $$\frac{dx}{dx} = 1$$

3) $$\frac{d(u+v)}{dx} = \frac{du}{dx} + \frac{dv}{dx}$$

4) $$\frac{d(uv)}{dx} = u\frac{dv}{dx} + v\frac{du}{dx}$$

```
(define derivada (lambda(exp var)
                   (cond ((number? exp) 0)
                         ((variable? exp) (if (misma-variable? exp var) 1 0))
                         ((suma? exp) (crea-suma
                                        (derivada (primeroperando exp) var)
                                        (derivada (segundooperando exp) var)))
                         ((producto? exp) (crea-suma
(crea-producto (primeroperando exp) (derivada (segundooperando exp) var))
(crea-producto (segundooperando exp) (derivada (primeroperando exp) var))))
                         (else (error "no se derivar esta expreson")))) )
```

EJEMPLOS

(derivada '(+ 3 x) 'x)

(+ 0 1)

(derivada '(+ x y) 'x)

(+ 1 0)

(derivada '(+ 3 x) 'x)

(derivada '(+ 3 x) 'x)

PROCEDIMIENTOS AUXILIARES

```
(define variable? (lambda(x) (symbol? x)))
```

```
(define misma-variable? (lambda(v1 v2)
                          (and (variable? v2) (eq? v1 v2))))
```

```
(define suma? (lambda(lista)
   (and (pair? lista) (eq? (car lista) '+))))
```

```
(define crea-suma (lambda(s1 s2)
                    (list '+ s1 s2)))
```

```
(define primeroperando (lambda(s)
                         (car (cdr s))))
```

```
(define segundooperando (lambda(s)
                          (car (cdr (cdr s)))))
```

```
(define producto? (lambda(lista)
       (and (pair? lista) (eq? (car lista) '*))))
```

```
(define crea-producto (lambda(s1 s2)
                        (list '* s1 s2)))
```

SIMPLIFICACIÓN

Si alguno de los operandos es 0 da el otro operando y si los dos son números se suman

```
(define crea-suma (lambda(s1 s2)
                    (cond ((=numero? s1 0) s2)
                          ((=numero? s2 0) s1)
                    ((and (number? s1) (number? s2)) (+ s1 s2))
                          (else'(+ s1 s2)))))
```

```
(define =numero? (lambda(exp num)
                   (and (number? exp) (= exp num))))
```

Si alguno de los operandos es 0 da 0
Si alguno de los operandos da 1 da el otro operando y si los dos son números se multiplican.

```
(define crea-producto (lambda(s1 s2)
    (cond ((or (=numero? s1 0) (=numero? s2 0)) 0)
                          ((=numero? s1 1) s2)
                          ((=numero? s2 1) s1)
                    ((and (number? s1) (number? s2)) (* s1 s2))
                          (else (list '* s1 s2)))))
```

REFERENCIAS

- HTTP://WWW.PLT-SCHEME.ORG/
- HTTP://MITPRESS.MIT.EDU/SICP/

OTRAS PLATAFORMAS

- [ ] HASKELL (HTTP://HASKELL.ORG/)

- [ ] MATHEMATICA
  (HTTP://DEMONSTRATIONS.WOLFRAM.COM/)

CONTACTO

Dr. José Manuel Gómez Soto
Laboratorio de Sistemas Complejos
Posgrado e Investigación
Universidad la Salle
México
www.ci.ulsa.mx/~jmgomez
jmgomezgoo@gmail.com