ALGUNAS FUNCIONES NO COMPUTABLES.-

Diremos que una función f : N $\rightarrow$ N es **computable** cuando existe un programa en P, Q, de índice q, tal que $\phi_q = f$

Consideremos la función θ : N $\rightarrow$ {0,1} definida por :

$\theta(n) = 1$ si $\langle Ix(n),n\rangle \downarrow$
$= 0$ si $\langle Ix(n),n\rangle \uparrow$

Teorema .- θ no es computable

Demostraremos este resultado por contradicción.

Supongamos que θ es computable. Existe entonces un programa Q que computa θ. Sea QM la macro de expresión asociada. Consideremos el siguiente programa *R* :

```
PROGRAM(X0)
     X1 := QM(X0);
     WHILE X1 ≠ 0 DO SKIP END
RESULT(X0)
```

(SKIP es una macro de sentencia que no afecta el estado).

Consideremos la propiedad de terminación de *R*

1) *R* diverge con entrada n si QM(n) da un valor distinto de 0.
Pero QM(n) $\neq$ 0 sii n representa un programa que converge con entrada n.
Por lo tanto :

$\langle R,n\rangle \uparrow$ si $\langle Ix(n),n\rangle \downarrow$

2) *R* converge con resultado 0 para la entrada n si QM(n) da el valor 0. Pero QM(n) = 0 sii n representa un programa que diverge con entrada n. Por lo tanto :

$\langle R,n\rangle \downarrow$ si $\langle Ix(n),n\rangle \uparrow$

Sea r el índice del programa R, Ix(r) = R

Consideremos que pasa con R cuando la entrada es r

1) $\langle R,r\rangle \uparrow$ si $\langle Ix(r),r\rangle \downarrow$, o sea si $\langle R,r\rangle \downarrow$

2) $\langle R,r\rangle \downarrow$ si $\langle Ix(r),r\rangle \uparrow$, o sea si $\langle R,r\rangle \uparrow$

Como un programa P para una entrada debe converger o diverger, hemos llegado a una contradicción y debemos descartar la única hipótesis : que θ es computable.

Por lo tanto, θ no es computable !!

Consideremos la siguiente función

stop (p,n) = 1 si <Ix(p),n>↓
0 en caso contrario

Teorema.- stop no es computable **(PROBLEMA de la PARADA)**

Demostraremos este teorema utilizando el resultado ya conocido de que θ no es computable. La técnica de reducir nuestro problema a un problema que ya sabemos que no es computable se usará muchas veces.

1) Supondremos que la función es computable
2) Utilizando el supuesto programa para computarla escribimos un programa que computa una función que ya sabemos no es computable.

Supongamos que stop es computable. Existe una macro MSTOP que computa la expresión asociada a la función stop. Podemos construir entonces el siguiente programa :

```
PROGRAM(X0)
     X1 := MSTOP(X0,X0)
RESULT(X1)
```

Pero este programa computa la función θ !!

Consideraremos una clase más restringida de programas y veremos si se siguen manteniendo los resultados sobre no computabilidad obtenidos hasta el momento. Se trata de los ***programas constantes***. Los definiremos como aquellos que no usan su variable de entrada.

Es claro que estos programas computan una función constante; si bien hay otros programas que pueden computar una función constante, sólo serán considerados los que no utilizan la variable de entrada en ninguna sentencia. Esta es una cuestión sintáctica acerca de un programa y es computable , la condición que se debe cumplir es :

cant-veces(x,prog(x,c,y)) = 1).

Antes de probar que la función stop restringida a programas constantes no es computable introduciremos la función

g : NxN $\rightarrow$ N t.q.

Dado q índice del programa Q $\equiv$

```
PROGRAM(Xi)
    C
RESULT(Xj)
```

y n, valor de su entrada, g(q,n) = q´ tal que

Ix(q´) es

```
PROGRAM(Xmax+1)
     Xi := n;
     C
RESULT(Xj)
```

O sea, g es una función que transforma programas en programas (índices de programas en índices de programas, por supuesto). El programa resultado es un programa constante de acuerdo a nuestra definición.

Notar que :
$\phi_{g(q,n)}(m) = \phi_q(n)$ para todo m natural

g es computable, sea GM la macro que la computa

```
EXPR GM(Y1,Y2)  MACRO
     Y3 := DECOD-PROG-VARE (Y1);
     Y4 := DECOD-PROG-SENT (Y1);
     Y5 := DECOD-PROG-VARS (Y1);
     Y6 := MAX-VAR-NUM (Y1);   Y6 := SUC(Y6);
     Y7 := COD-EXPR-0;
     Y7 := COD-ASS(Y3,Y7);
```

```
        IF Y2 ≠ 0 THEN
             Y8 := COD-SUC(Y3);
             Y8 := COD-ASS(Y3,Y8);
             Y9 := COD-SEC(Y8,Y4);
             Y2 := PRED(Y2);
             WHILE Y2 ≠ 0 DO
                  Y9 := COD-SEC(Y8,Y9);
                  Y2 := PRED(Y2)
             END;
             Y9 := COD-SEC(Y7,Y9)
        ELSE
             Y9 := COD-SEC(Y7,Y4)
        FI;
        Y10 := COD-PROG(Y6,Y10,Y5)
    RESULT(Y10)
```

Teorema .- **(PROBLEMA de la PARADA para PROGRAMAS CONSTANTES)**

stopK(p) = 1 si Ix(p) es un programa cte y <Ix(p),m>$\downarrow \forall$ m

0 en caso contrario

no es computable.

Dem.- Supongamos que stopK es computable; sea MSK la macro que la computa. Utilizando además la macro GM podemos construir la siguiente macro :

```
EXPR Q(Y0,Y1) MACRO
        Y2 := GM(Y0,Y1);
        Y2 := MSK(Y2)
RESULT(Y2)
```

Pero esta macro computa stop(p,n). Dado que la función g es computable, como lo demuestra la macro GM que hemos construido, debemos descartar la existencia de MSK, o sea, stopK no es computable.

EL PROBLEMA DEL ***CÓDIGO MUERTO***.-

Supongamos que en un programa tenemos :

...
...

```
X1 := 0;
IF X1 ≠ 0 THEN C FI;
...
```

Sabemos que el conjunto de sentencias C nunca será ejecutado.

¿ Es posible para un computador averiguar si existe ***código muerto*** en un programa?

Para representar adecuadamente este problema numeraremos las sentencias del programa, p.ej.

```
1       X := ..
2       WHILE ...
3               X1 := ..
4               X2 := ..
        END;
5       X6 := ..
        .
        .
```

Definamos el predicado

ejec-sent-num (p,n,m) verdadero si la sentencia número m es ejecutada alguna vez en la computación de Ix(p) con entrada n

y la función

δ (p,m)	=	1	si ∃ n / ejec-sent (p,n,m) verdadero
		0	en caso contrario

Entonces, si δ (p,m) = 0, la sentencia m es código muerto en Ix(p).

Nos interesa saber si δ (p,m) es computable.

Veremos primero que pasa con un caso más simple :

δ'(p,n,m)	=	1	si ejec-sent-num (p,n,m) verdadero
		0	en caso contrario

Esta función no es computable.

Consideremos una macro PM , que dado un programa Q con índice **q** y **max** como número más alto de variable :

```
PROGRAM(Xi)
        C
```

RESULT(Xj)

calcula el par <p,m>, donde p es el índice del programa :

```
PROGRAM(Xi)
        C;
        Xmax+1 := 0
RESULT(Xj)
```

y m el número de la última sentencia de este programa.

(Ejercicio, construir la macro PM).

Supongamos que δ' es computable. Sea MDELTA una macro que la computa. Podemos construir el siguiente programa :

```
PROGRAM(X0)                 XO = <q,n>
     X1 := FST(X0);         q
     X2 := SND(X0);         n
     X3 := PM(X1);          <p,m>
     X4 := FST(X3);         p
     X5 := SND(X3);         m
     X6 := MDELTA(X4,X2,X5)
RESULT(X6)
```

Este programa computa stop(q,n) !!

(Notar que los programas en P terminan siempre su ejecución en la última sentencia sintáctica del programa y que se cumple:

<Ix(q),n> $\downarrow$ sii Ix(p) ejecuta la sentencia m al computar con entrada n)

Ejercicio.- Demostrar que δ (p,m) no es computable.

EQUIVALENCIA DE PROGRAMAS

Otra cuestión de interés práctico es si existe la posibilidad de saber si dos programas computan la misma función.

Trataremos primero un caso más simple.

Consideremos la función :

$\mathbf{eq_{id}(x)}$ = 1 si ϕ_X = id (la identidad)
0 en caso contrario

Teorema - eq_{id} no es computable.

Dem.- Dado un programa q y un número n podemos computar el índice del programa Q :

```
PROGRAM(X0)
     X1 := EVAL-PROG(q,n)
RESULT(X0)
```

Q computa la identidad sii $<Ix(q),n> \downarrow$

Supongamos que eq_{id} es computable, sea MID una macro que la computa. Sea MQ una macro que dados q y n calcula el índice del programa Q anterior. Podemos construir :

```
PROGRAM(<X0,X1>)
     X2 := MQ(X0,X1);
     X3 := MID(X2)
RESULT(X3)
```

Este programa computa stop (q,n) !!

Teorema - No existe un programa que pueda decidir si dos programas arbitrarios computan la misma función.

Dem.- Ejercicio . Se sugiere prueba por contradicción utilizando el teorema 10.

UNA FUNCIÓN COMPUTABLE :

Necesitamos eval-prog-step para resolver algunos problemas "semidecidibles". Consideremos la función parcial :

$f(q,r,n) = 1$ si $\phi_q(n) = 1$ o $\phi_r(n) = 1$
indefinida en caso contrario

Si tratamos de usar eval-prog para computar esta función, tendremos el problema de que la computación puede no terminar para el primero de los programas que se evalúa, en un caso en que sí termina el segundo.

Resolveremos este problema mediante el siguiente programa:

```
PROGRAM(X0)
     X1 := FST(X0);       q
     X2 := SND(X0);       <r,n>
     X3 := SND(X2);       n
     X2 := FST(X2);       r
     X4 := 1;  X5 := 1;
     WHILE X4 ≠ 0 DO
          X6 := EVAL-PROG-STEP(X1,X3,X5);
          IF SND(X6) = 1 THEN X4 := 0 FI;
          X6 := EVAL-PROG-STEP(X2,X3,X5);
          IF SND(X6) = 1 THEN X4 := 0 FI;
          X5 := SUC(X5)
     END;
     X4 := 1
RESULT(X4)
```

Práctico

Ejercicio 1.-
Demostrar que :

$\rho(p,m) = 1$ si $\exists n$ /ejec-sent-num (p,n,m) = 1
0 en caso contrario

no es computable.

(ejec-sent-num (p,n,m) = 1 si la sentencia número **m** es ejecutada alguna vez en la computación de Ix(p) con entrada **n**
0 en caso contrario)

Ejercicio 2.-
Demostrar que no existe un programa que pueda decidir si dos programas arbitrarios computan la misma función.

Ejercicio 3.-
Sea $f : NxN \rightarrow N$ una función total

Decimos que $g : N \rightarrow N$ viene de f por ***miniminalización***, notación $g(x) = \mu y(f(x.y)=0)$, cuando :

$g(x) = y$ sii $f(x,y) = 0$ y $\forall z < y$, $f(x,z) \neq 0$

Demostrar que si f es computable, también lo es g.

Ejercicio 4.-

¿ Son computables las siguientes funciones? Demostrar.

a) $K(n) = 1$ si $\langle Ix(n), n\rangle \downarrow$
indef. en caso contrario

b) $cstop(p,n) = 1$ si $\langle Ix(p), n\rangle \uparrow$
indef. en caso contrario