# Práctica IV: Codificación de vectores y programas WHILE en Octave

## Teoría de Autómatas y Lenguajes Formales

**Ejercicio 1. Cree el programa WHILE más simple que calcule la función "diverge" (con cero argumentos)**

diverge = (0, s)

s:

$X_2$:= $X_1 + 1$ (*Nos aseguramos de que $X_2 \neq 0$*)
**while** $X_2 \neq 0$ **do**
*(*Bucle infinito*)*
$X_1$:= *0*
**od**

Podemos comprobar en Octave que su codificación es:

```
>> WHILE2N(0, "X2:=X1+1;while X2!=0 do X1:=0 od")
ans = 59160002
>> CODE2N("X2:=X1+1;while X2!=0 do X1:=0 od")
ans = 10876
```

*Codificación mínima de la función "diverge"*

**Ejercicio 2. Describe en Octave una función que enumere todos los vectores de cualquier dimensión**

Ya tenemos un script que devuelve el vector asociado a un número natural. Esto simplifica mucho las cosas, ya que podemos usarlo para obtener todos los vectores: $godelencoding(1), godelencoding(2), \ldots$. *godelencoding* está definido en los apuntes como $\gamma$, cuyo segundo argumento define la componente del vector que queremos devolver. Definimos, pues, la función $godelall : \mathbb{N} \to P(\mathbb{N}^*)$ que, dado un natural (o el 0), devuelve los vectores asociados a todos los naturales hasta ese número. Nótese que la función no es sobreyectiva, ya que $|P(\mathbb{N}^*)| = \aleph_1 \neq \aleph_0 = |\mathbb{N}^*|$

```
function code=godelall(n)
   encodings=["(",int2str(godeldecoding(0)),")"]
   for i = 1:n
     encodings=[encodings,", (",int2str(godeldecoding(i)),")"]
   endfor
end
```

Podemos comprobar, al ejecutar y comparar en los apuntes, que el programa funciona:

```
>> godelall(10)
encodings = ()
encodings = (), (0)
encodings = (), (0), (0  0)
encodings = (), (0), (0  0), (1)
encodings = (), (0), (0  0), (1), (0  0  0)
encodings = (), (0), (0  0), (1), (0  0  0), (1  0)
encodings = (), (0), (0  0), (1), (0  0  0), (1  0), (2)
encodings = (), (0), (0  0), (1), (0  0  0), (1  0), (2), (0  0  0  0)
encodings = (), (0), (0  0), (1), (0  0  0), (1  0), (2), (0  0  0  0), (1  0  0)
encodings = (), (0), (0  0), (1), (0  0  0), (1  0), (2), (0  0  0  0), (1  0  0), (0  1)
encodings = (), (0), (0  0), (1), (0  0  0), (1  0), (2), (0  0  0  0), (1  0  0), (0  1), (3)
```

*Ejemplo exitoso de ejecución*

**Ejercicio 3. Describe en Octave una función que enumere todos los programas WHILE**

Ya tenemos un script que devuelve el código WHILE asociado a un número natural. Esto simplifica mucho las cosas, ya que podemos usarlo para devolver todos los programas WHILE: $N2WHILE(1), N2WHILE(2), \ldots$. $N2WHILE$ está definido en los apuntes. Definimos, pues, la función $whileall : \mathbb{N} \to P(WHILE)$ que, dado un natural (o el 0), devuelve los programas asociados a todos los naturales hasta ese número. De nuevo, la función no es sobreyectiva.

```
function code=whileall(n)
encodings=N2WHILE(0)
   for i = 1:n
     encodings=[encodings,"\n#######\n",N2WHILE(i)]
   endfor
end
```

Podemos usar esta función para comprobar que el ejercicio 1 está bien resuelto, introduciendo su codificación y viendo si antes de que aparezca nuestro programa,aparece otro que también diverja. En la práctica, no podemos hacer eso, ya que la codificación es un número muy alto.

```
>> WHILEall(10)
(0, X1=0)
###########
(1, X1=0)
###########
(0, X1=0; X1=0)
###########
(2, X1=0)
###########
(1, X1=0; X1=0)
###########
(0, X1=X1)
###########
(3, X1=0)
###########
(2, X1=0; X1=0)
###########
(1, X1=X1)
###########
(0, X1=0; X1=0; X1=0)
###########
(4, X1=0)
###########
```

*Programas WHILE de codificación menor o igual que 10*