Anexo

En este anexo mostraremos varios ejemplos de máquinas de Turing. Para cada una de ellas daremos su diagrama de estados y una implementación en Python. Es recomendable ir observando el diagrama ya que ayuda a comprender el funcionamiento de las máquinas.

En primer lugar notar que los estados por si solos no hacen nada. Es necesario tener un soporte (una máquina) donde introducir los estados para poder ejecutarlos. A continuación incluiremos la clase "Máquina de Turing" sobre la que hemos construido los estados para las distintas máquinas.

```
class MáquinaTuring():
    def __init__(self, cintaInicial, estados, estadosFinales=[], intervaloT=[0,0],
    pasoApaso=False) :

        self.Cinta = cintaInicial


        self.estados = estados
        self.estadosFinales = estadosFinales
        self.intervaloT = intervaloT
        self.pasoApaso = pasoApaso
        self.iActual = 0

        self.ventana = soporteGráfico.Ventana(self)
        self.ventana.muestraLista(self.Cinta)

        self.evolución()


    def evolución(self):
        q0 = self.estados[0]
        estadosSucesivos = [q0]
        i = 0
        i0 = 0
        print('evolución', self.Cinta)

        while  q0 not in self.estadosFinales and i < 450:
            if not (q0 in estadosSucesivos):
                estadosSucesivos.append(q0)


            print(q0.__name__, " '",self.Cinta[i],"' ",i,sep='')
            i, q = q0(self.Cinta, i0)
            self.iActual = i
```

```
        if q==q0:
            tIntervalo = self.intervaloT[0]
        else:
            tIntervalo = self.intervaloT[1]

        if (i>= i0):
            self.ventana.completaLista(self.Cinta,i0)

        if (i0 != i): self.ventana.pintaRectángulo(self.Cinta, i0, 0)
        self.ventana.pintaRectángulo(self.Cinta, i, 1)

        if self.pasoApaso:
            time.sleep (tIntervalo)
            self.ventana.master.update()

        q0 = q
        i0 = i
    print(q0.__name__)
    print (self.Cinta)
    print ('# estados =', len(estadosSucesivos))
    self.ventana.mainloop()
```

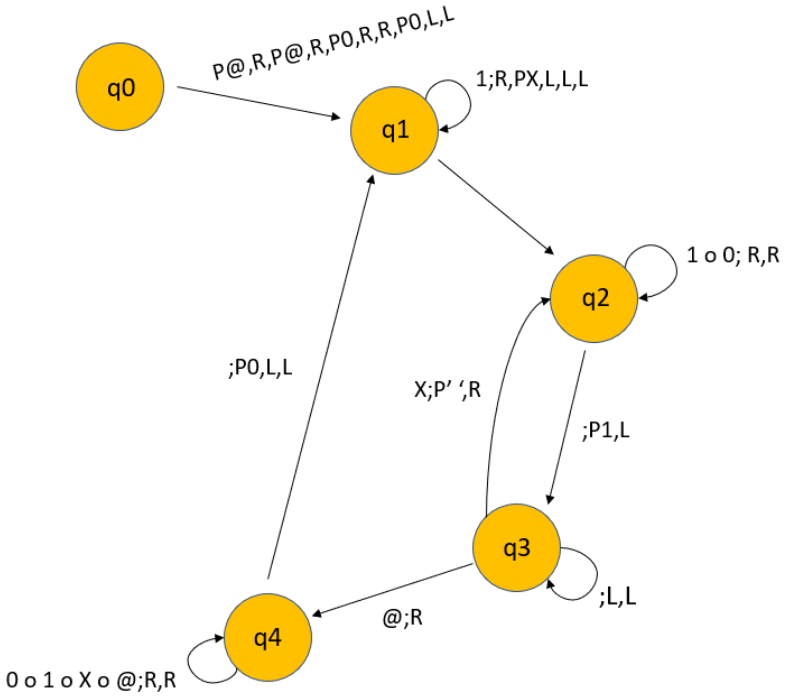
.1. Estados máquina que escribe un número irracional

.1.1. Explicación

Vamos a ver el funcionamiento de la máquina que computa la secuencia 001011011101111... que representa un número irracional. Este es el segundo ejemplo de máquina presentado por Turing en su artículo. Al tratarse de una secuencia infinita estaremos ante una máquina no circular.

La máquina va a constar de cinco estados, $q0, q1, q2, q3, q4$, donde $q0$ va a ser el estado inicial y no vamos a tener estado final ya que hemos visto que va a ser una máquina no circular. El estado $q0$ va a escribir dos símbolos "@" consecutivos para marcar el inicio del cómputo y a continuación va a escribir los dos primeros ceros de la secuencia. Una vez hecho esto va a pasar al estado $q1$. Lo que va a hacer este estado es marcar con una x los 1 que haya en la cinta. Si no encuentra ningún 1 o ya ha marcado todos pasa a $q2$. $q2$ va a recorrer la secuencia ya escrita de 0 y 1 hasta llegar al final, escribirá un 1 y pasará a $q3$. $q3$ se va a encargar de comprobar si hay alguna x en la cinta recorriéndola de derecha a izquierda. Si encuentra alguna pasará a $q2$ para escribir un 1 al final de la cinta y lo hará tantas veces como x haya, y las irá eliminando. Una vez que no queden x y haya llegado al inicio de la cinta pasará a $q4$, y lo que hará $q4$ será ir hasta el final de la cinta, escribir un 0 y pasar a $q1$ para volver a empezar el algoritmo.

.1.2. Diagrama



.1.3. Código

```
def q0(cinta, i):
    cinta[i] = 'a'
    i = __R__(cinta,i)
    cinta[i] = 'a'
    i = __R__(cinta,i)
    cinta[i] = '0'
    i = __R__(cinta,i)
    i = __R__(cinta, i)
    cinta[i] = '0'
    i = i-2
    return i, q1


def q1(cinta, i):
    if cinta[i] == '1':
        i = i+1
        cinta[i] = 'x'
        i = i-3
        return i, q1
    else:
        return i,q2


def q2(cinta, i):
    if cinta[i] == '1' or cinta[i] == '0':
        i = __R__(cinta, i)
        i = __R__(cinta, i)
        return i, q2
```

```
    else:
        cinta[i] = '1'
        i = i-1
        return i, q3


def q3(cinta, i):
    if cinta[i] == 'x':
        cinta[i] = ' '
        i = i+1
        return i, q2
    elif cinta[i] == 'a':
        i = i+1
        return i, q4
    else:
        i= i-2
        return i, q3


def q4(cinta, i):
    if cinta[i] == '0' or cinta[i] == '1' or cinta[i] == 'x' or cinta[i] == 'a':
        i = __R__(cinta, i)
        i = __R__(cinta, i)
        return i, q4
    else:
        cinta[i] = '0'
        i = i-2
        return i, q1
```

.2. Estados máquina que invierte un número

.2.1. Explicación

Vamos a ver cuál sería el funcionamiento de una máquina de Turing que invierte un número. En primer lugar notar que, a diferencia de los ejemplos vistos hasta ahora, la cinta no está inicialmente vacía sino que contiene la secuencia que deseamos invertir. En este tipo de casos se escriben dos símbolos "@" consecutivos para indicar el inicio del problema y a continuación se introduce el número que en este caso vamos a invertir respetando las celdas pares e impares. La cinta se vería algo así:

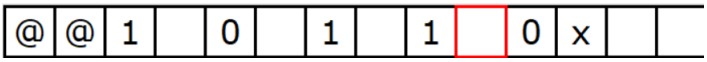
@	@	1		0		1		1		0	

Cabe destacar que para generar esta máquina hemos empleado distintas tablas esqueleto y hemos condensado varios movimientos en un único estado para evitar estados redundantes. Se podría hacer según la estructura de instrucciones vista a lo largo del trabajo (en función de un estado y el símbolo leído se escribe un símbolo, se realiza un desplazamiento de una única posición y pasamos a otro estado) pero el código sería muy largo y muy repetitivo y no sería óptimo. Ahora vamos a describir los pasos que hemos seguido.

Una vez que hemos introducido el número a invertir la máquina empieza a trabajar. Notar que este programa tiene un final ya que cuando hayamos invertido el número el proceso debe terminar, por tanto

es una máquina circular y va a necesitar de, al menos, un estado final. En este caso hemos necesitado solo uno que es q_final. En el momento en que lleguemos a este estado el programa se detendrá.

Empezamos pues en el estado inicial que es $q0$. El cabezal de la máquina empezará situado en la primera "@" y lo que hará $q0$ será avanzar dos posiciones a la derecha para situarse al principio del número. Ua vez que se haya desplazado pasará al estado $q\_decisin\_final\_1$. Este estado es una mera comporbación de que después de la segunda "@" tenemos un 0 o un 1. En caso de que haya un símbolo disntinto (por ejemplo un espacio en blanco si no se ha respetado correctamente el orden de celdas pares e impares) pasaríamos al estado q_final y el programa se detendría. Como tenemos un 1 pasamos al estado $q1$. El estado $q1$ lo que hará es encontrar el último dígito no marcado del número, donde por dígito marcado entenderemos aquel que tenga una x situada a su derecha en la celda más próxima (celda impar). Una vez que $q1$ encuentre ese dígito lo marcará y este será el primer número que vamos a cambiar y pasará al estado $q\_decisin\_final\_2$. La cinta en este momento se vería así:

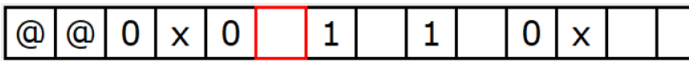


$q\_decisin\_final\_2$ sirve para comporbar si el programa debe finalizar ya. Si la celda que está en rojo tuviese una x significaría que ya hemos finalizado la inversión y pasaría al estado final. Como no es el caso nos desplazamos una posición a la derecha y pasamos a $q2$.

Ahora lo que tendremos que hacer es llevar nuestro dígito marcado a la izquierda y coger su correspondiente dígito y llevarlo a la derecha ya que una forma de invertir un número es intercambiando las cifras que equidisten del dígito intermedio del número, como si fuesen parejas. Pero no es lo mismo llevar a la izquierda un 0 o un 1 y eso es lo que hará $q2$: nos diferenciará si hemos marcado un 0 o un 1. Una vez hecha esta diferencia, si lo que hemos marcado era un 0 pasaremos al estado $llevaIzda('0')$ y si era un 1 pasaremos a $llevaIzda('1')$. En ambos casos lo que haremos será buscar el dígito situado más a la izquierza que no esté marcado y una vez encontrado haremos lo siguiente:

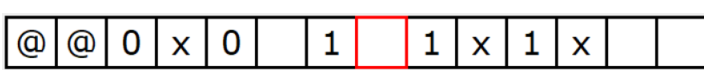
- Primero diferenciaremos si el dígito es un 0 o un 1.
- Luego escribiremos el dígito que traíamos de antes, en este caso un 0.
- Después de haber escrito el dígito marcaremos esa celda con una x para saber que ya hemos realizado una inversión.
- Por último iremos al estado $llevaoDcha$ o $lleva1Dcha$ en función de si antes había un 0 o un 1.

La cinta hasta este momento se vería así:



Ahora lo último que nos queda por hacer es llevar el dígito de la izquierda a la derecha. Como en este caso había un 1 hemos pasado al estado $lleva1Dcha$ y este lo que hará es buscar la x que le indique dónde está la celda donde tiene que escribir ese 1. Una vez encontrada la celda escribiremos el 1, marcaremos el siguiente dígito a invertir y pasaremos de nuevo al estado $q\_decisin\_final\_2$ para realizar la comporbación de si ya hemos acabado, y si no es el caso volveremos a repetir el proceso hasta finalizar.

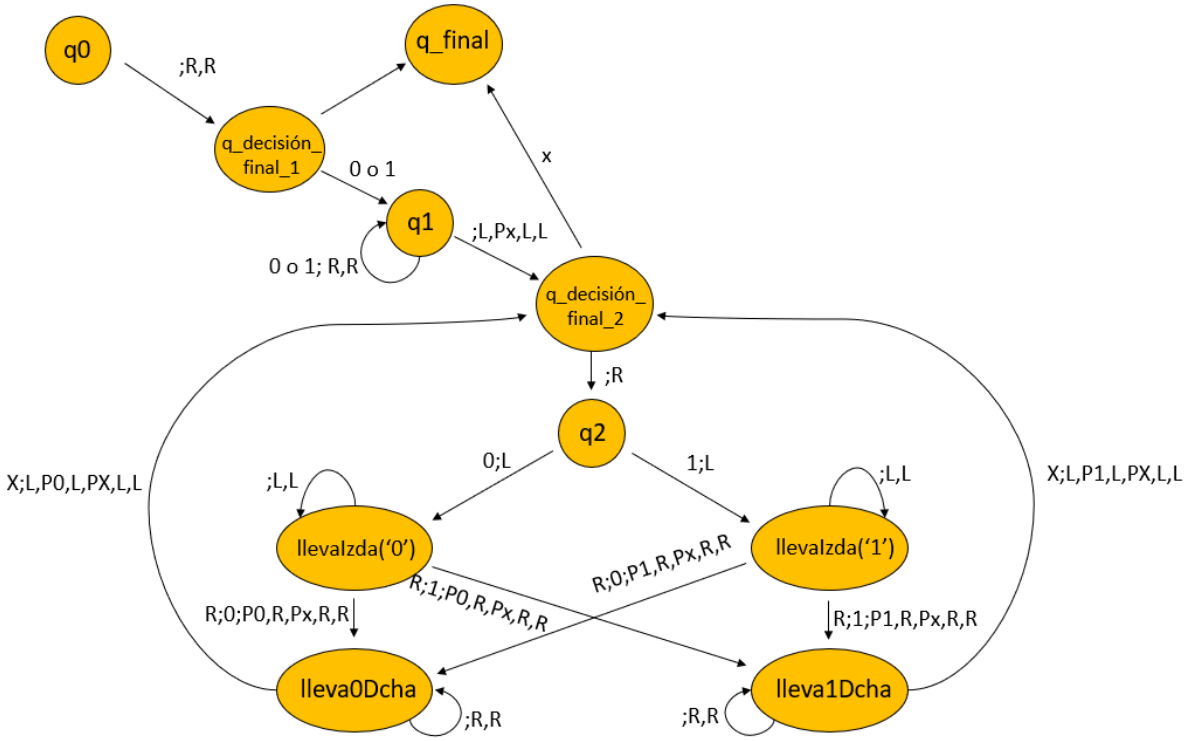
Después de realizar el primer bucle de inversión la cinta se vería así:



Y el resultado final sería el siguiente:

@	@	0	x	1	**x**	1	x	0	x	1	x		

.2.2. Diagrama



.2.3. Código

```
def q0(cinta,i):
    i = i+2
    return i, q_decision_final_1

def q_decision_final_1(cinta,i):
    if cinta[i] == ' ' or cinta[i] == 'x':
       return i, q_final
    else:
        return i, q1

def q_final(cinta,i):
    return i, q_final

def q1(cinta,i):
    if cinta[i] == '1' or cinta[i] == '0':
        i = i+2
        q = q1
    else:
        i=i-1
        cinta[i] = 'x'
```

```
        i=i-2
        q= q_decision_final_2
    return i, q

def q_decision_final_2(cinta,i):
    if cinta[i] == ' ':
        i=i+1
        return i, q2
    else:
        return i, q_final

def q2(cinta,i):
    if cinta[i]== '0':
        i = i-1
        return i, q3
    else:
        i = i-1
        return i, q4

def llevaDerecha(C, d):
    def llevaDcha(cinta,i):
        if cinta[i] == ' ':
            i = i+2
            return i, llevaDcha
        else:
            i = i-1
            cinta[i] = d
            i = i-1
            cinta[i] = 'x'
            i = i-2
            return i, C
    return llevaDcha

def llevaIzda (d):
    def buscaPrimerDígito(cinta,i): #q3
        if cinta[i] == ' ':
            i = i-2
            return i, buscaPrimerDígito
        else:
            i = i+1
            return i, escribePrimerDígito

    def escribePrimerDígito(cinta, i): #q33
        if cinta[i] == '0':
            cinta[i] = d
            i = i+1
            cinta[i] = 'x'
            i = i+2
            return i, llevaODcha
        else:
            cinta[i] = d
```

```
            i = i + 1
            cinta[i] = 'x'
            i = i + 2
            return i, lleva1Dcha

    return buscaPrimerDígito

q3 = llevaIzda('0')
q4 = llevaIzda('1')
lleva0Dcha = llevaDerecha(q_decision_final_2, '0')
lleva1Dcha = llevaDerecha(q_decision_final_2,'1')

listaEstados=[q0]
estadosFinales=[q_final]
```

.3. Estados máquina que suma dos números en bianrio

.3.1. Explicación

No daremos una explicación detallada de esta máquina ya que sería bastante extensa pero sí explicaremos a grandes rasgos su funcionamiento. En el código describiremos las tablas esqueleto empleadas en esta máquina ya que dan una visión más intuitiva de su funcionamiento y después pondremos los estados de forma detallada.

En la cinta tendremos dos números en binario separados por un signo '+' y después del último dígito habrá un '='.

@	@		1		1		0		+		1		1		1		=			

En primer lugar marcará con una x el primer número.

@	@		1	x	1	x	0	x	+		1		1		1		=			

Después marcará con una y el segundo número.

@	@		1	x	1	x	0	x	+		1	y	1	y	1	y	=			

Una vez marcados ambos números copiará el primero a la derecha del igual. Notar que lo copiará invertido.

@	@		1		1		0		+		1	y	1	y	1	y	=		0		1		1		

A continuación sumará el número situado a la izquierda del igual con el que hemos trasladado al final de la lista.

@	@		1		1		0		+		1		1		1		=	Q	1		0		1		1		

Por último inveritá el número. Y esto sabemos como hacerlo ya que antes hemos creado una máquina que invierte una cadena de símbolos en binario.

.3.2. Código

Expresaremos el código de dos formas alternativas. En esta primera la ejecución se realiza paso a paso y es más intuitiva.

```
Tablas esqueleto:

inv = inversion()
sumaY = suma (inv, mSumando='y', mSuma='=')
copiaX = copia (sumaY, marcaNumero='x', posicion='=')
marcaY = marca (copiaX, qFinal, letra='y',signo='=')
marcaX = marca (marcaY, qFinal, letra='x',signo='+')
```

En esta segunda se ejecuta todo en una única sentencia. Esto pone de relieve el caracter recursivo de los estados de las máquinas de Turing.

```
marcaX = marca (   marca (   copia (   suma (   inversion(),
mSumando='y', mSuma='='),
                                        marcaNumero='x', posicion='='
                                    ),
                                qFinal, letra='y',signo='='
                            ),
                        qFinal, letra='x',signo='+'
                )
```

```
Estados:

def buscaÚltimo(C, B, **kwargs):
    def f(cinta, i):  # busca la primera aparición del carácter a
        nonlocal C, B, kwargs
        if cinta[i] == ' ':
            q = f1
        else:
            q = f
        i = __R__(cinta, i)
        return i, q

    def f1(cinta, i):
        nonlocal C, B, kwargs
        if cinta[i] == ' ':
            q = f2
        else:
            q = f
        return i, q
```

```
    def f2(cinta, i):
        nonlocal C, B, kwargs
        if cinta[i] == kwargs['valorBuscado']:
            q = C
        elif cinta[i] == '@':
            q = B
        else:
            i = i - 1
            q = f2
        return i, q

    mensaje = ''
    if 'valorBuscado' in kwargs.keys():
        mensaje = mensaje + 'busca=' + kwargs['valorBuscado'] + '\t'
    if 'valorGuardado' in kwargs.keys():
        mensaje = mensaje + 'lleva=' + kwargs['valorGuardado']
    f.__name__ = 'buscaÚltimo.' + f.__name__ + '( ' + mensaje + ' )'
    f1.__name__ = 'buscaÚltimo.' + f1.__name__ + '( ' + mensaje + ' )'
    f2.__name__ = 'buscaÚltimo.' + f2.__name__ + '( ' + mensaje + ' )'
    return f

"""marca con una letra (kwargs['letra']) el segmento situado
inmediatamente a la izquierda del primer signo
indicado ( kwargs['signo'])"""
def marca (C, B, **kwargs):
    def mueveIzda(cinta, i):
        return i-2, decide
    def decide(cinta, i):
        if cinta[i] in ['0','1']:
            q = marca
            i = __R__(cinta, i)
        else:
            q = C
        return i, q
    def marca (cinta, i):
        nonlocal C, kwargs
        cinta[i] = kwargs['letra']
        return i-3, decide

    bP = buscaPrimero (mueveIzda, B, kwargs['signo'])
    return bP
def marcaQZ(C,B,signo):
    def D2 (cinta, i):
        i = __R__(cinta, i)
        return i, escribeQ
    def escribeQ(cinta, i):
        cinta[i] = 'Q'
        i = __R__(cinta, i)
        return i, añade0
    def añade0(cinta, i):
        cinta[i] = '0'
```

```
        i = __R__(cinta, i)
        return i, marcaZ
    def marcaZ(cinta, i):
        cinta[i] = 'z'
        return i, C
    return buscaPrimero (D2, B, signo)

""" Añade el número marcado con 'mSumando' al número marcado con 'mSuma' """
def suma (C, **kwargs):

    def sumaLlevada(cinta, i):
        if cinta[i] in [' ', '0']:
            cinta[i] = '1'
            q = bPzFinal
        else:
            cinta[i] = '0'
            i = __R__(cinta, i)
            i = __R__(cinta, i)
            q = sumaLlevada
        return i, q

    def borraYL(cinta, i):
        cinta[i] = ' '
        return i - 1, iniciaSuma

    def iniciaSuma(cinta, i):
        if cinta[i] == '0':
            q = bPz0
        else:
            q = bPz1
        return i, q

    def preSuma1(cinta, i):
        return i - 1, suma1

    def suma1(cinta, i):
        if cinta[i] in [' ', '0']:
            cinta[i] = '1'
            i = __R__(cinta, i)
            q = avanzaDígito
        else:
            cinta[i] = '0'
            i = __R__(cinta, i)
            i = __R__(cinta, i)
            q = sumaLlevada
        return i, q

    def avanzaDígito(cinta, i):  # Borra y devuelve C
        cinta[i]=' '
        return i+1, pon0
```

```
    def pon0 (cinta, i):
        if cinta[i]==' ':
            cinta[i]='0'
        i = __R__(cinta, i)
        return i, añadeZ

    def añadeZ (cinta, i):
        cinta[i] =kwargs['mSuma']
        return i, bU

    def borraZ (cinta, i):
        cinta[i] = ' '
        return i-1, limpia0

    def limpia0 (cinta, i):
        if cinta[i] == '0':
            cinta[i]=' '
            i = i-2
            q= limpia0
        else:
            q = C
        return i, q

    preborraZ = buscaÚltimo( borraZ, C, valorBuscado=kwargs['mSuma'])
    bPzFinal = buscaÚltimo(avanzaDígito, qFinal, valorBuscado=kwargs['mSuma'])
    bPz1 = buscaPrimero(preSuma1, qFinal, kwargs['mSuma'])
    bPz0 = buscaPrimero(avanzaDígito, qFinal, kwargs['mSuma'])

    bU = buscaÚltimo (borraYL, preborraZ, valorBuscado=kwargs['mSumando'])

    return bU

def invierte ():
    # Skeleton tables:
    def llevaIzda(B, **kwargs):
        def Left(cinta, i):
            print('\n\n\n\nLlevo', kwargs['letra'])
            cinta[i] = ' '
            return i - 1, escribe

        def escribe(cinta, i):
            if cinta[i] == '0':
                q = llevaDcha0
            else:
                q = llevaDcha1
            cinta[i] = kwargs['letra']
            return i, q

        bP = buscaPrimero(Left, B, kwargs['signo'])
        return bP
    def llevaDcha(B, **kwargs):
```

```
        def Left(cinta, i):
            cinta[i] = ' '
            return i - 1, escribe

        def escribe(cinta, i):
            cinta[i] = kwargs['letra']
            return i - 2, reinicia

        def reinicia(cinta, i):
            if cinta[i] == '0':
                q = llevaIzda0
            else:
                q = llevaIzda1
            return i, q

        bP = buscaÚltimo(Left, B, valorBuscado=kwargs['signo'])
        return bP

    #Estados
    def Left (cinta, i):
        return i-1, iniciaInversion
    def iniciaInversion (cinta, i):
        if cinta[i]=='0':
            q = llevaIzda0
        else:
            q = llevaIzda1
        return i, q

    llevaDcha0 = llevaDcha (qFinal,  letra='0',signo='x')
    llevaDcha1 = llevaDcha (qFinal,  letra='1',signo='x')
    llevaIzda0 = llevaIzda (qFinal,  letra='0',signo='x')
    llevaIzda1 = llevaIzda (qFinal,  letra='1',signo='x')

    bU = buscaÚltimo(Left, qFinal,valorBuscado= 'x')

    marcaX = marca(bU, qFinal, letra='x',signo='F')

    return marcaX

# Estados:
def qFinal(cinta, i):
    return i, qFinal


# Preparan el segundo sumando
def añadeZ (cinta, i):
    i = __R__(cinta,i)
    i = __R__(cinta, i)
    return i, escribeZ
def escribeZ (cinta, i):
    cinta[i] = 'z'
```

```
    return i, estadosSumaY
bUQ = buscaÚltimo (añadeZ, qFinal, valorBuscado='Q')
# Dejan F final
def inversion():
    def RR (cinta, i):
        i = __R__(cinta,i)
        i = __R__(cinta,i)
        return i, marcaF
    def marcaF (cinta, i):
        cinta[i]='F'
        return i, invierte()
    return RR
```

.4. Estados Máquina Universal de Turing

Recordemos que la Máquina Universal está compuesta por 9 tablas esqueleto principales y varias auxiliares. En primer lugar pondremos las tablas auxiliares y después las que definen la Máquina Universal.

Tanto la notación como la descripción de los estados los hemos sacado del propio artículo de Turing [5] y del apéndice del libro *A science of Operations* [8].

```
def config (C, a):
    def con(cinta, i):
        if cinta[i] !='A':
            i = __R__ (cinta,i)
            i = __R__(cinta, i)
            q = con
        else:
            i = i - 1
            cinta[i] = a
            i = __R__ (cinta, i)
            q = con1
        return i, q

    def con1 (cinta, i):
        if cinta[i]=='A':
            q = con1
        else:
            q = con2
        i = __R__(cinta, i)
        cinta[i] = a
        i = __R__(cinta, i)
        return i, q

    def con2 (cinta, i):
        if cinta[i]=='C':
            i = __R__(cinta, i)
            cinta[i] = a
            i = __R__(cinta, i)
            q = con2
        else:
            i = __R__(cinta, i)
```

```
            i = __R__(cinta, i)
            q = C
        return i, q

    return con


def buscaPrimero(C, B, a):
    def f(cinta, i):  # busca la primera aparición del carácter a
        if cinta[i] == '@':
            i = i - 1
            q = f1
        else:
            i = i - 1
            q = f
        return i, q

    def f1(cinta, i):
        if cinta[i] == a:
            q = C
        elif cinta[i] == ' ':
            i = __R__(cinta, i)
            q = f2
        else:
            i = __R__(cinta, i)
            q = f1
        return i, q

    def f2(cinta, i):
        if cinta[i] == a:
            q = C
        elif cinta[i] == ' ':
            i = __R__(cinta, i)
            q = B
        else:
            i = __R__(cinta, i)
            q = f1
        return i, q

    f.__name__ = 'buscaPrimero.' + f.__name__ + "( '" + a + "' )"
    f1.__name__ = 'buscaPrimero.' + f1.__name__ + "( '" + a + "' )"
    f2.__name__ = 'buscaPrimero.' + f2.__name__ + "( '" + a + "' )"

    return f

# erase
def borra (C, B, alpha):
    def e1(cinta, i):
        print ('Borro')
        cinta[i] = ' '
        return i, C
```

```
    return buscaPrimero (e1, B, alpha)

#e
def borraTodo (B, alpha):
    def e (cinta, i):
        return i, aux

    aux = borra (e, B, alpha)
    return e


# pe
def escribeAlFinal (C, beta):
    def pe (cinta, i):
        return i, buscaPrimero (pe1, C, '@')

    def pe1 (cinta, i):
        if cinta[i]!= ' ':
            i = __R__ (cinta, i)
            i = __R__(cinta, i)
            q = pe1
        else:
            cinta[i] = beta
            q = C
        return i, q

    return pe

# move left
def izquierda (C):
    def mI (cinta, i):
        i = i-1
        return i, C
    return mI


# f'
def buscaMueveIzda (C, B, a):
    def BMI (cinta, i):
        return i, buscaPrimero( izquierda(C), B, a)
    return BMI

# copy
def copia (C, B, a):
    def c(cinta, i):
        return i, buscaMueveIzda(c1, B, a)

    def c1(cinta, i):
        return i, escribeAlFinal(C, cinta[i])

    return c
```

```
def copiaYBorra (C,B, a):
    def c (cinta, i):
        return i, copia(borra(C, B, a), B, a)
    return c

# copia y borra todo
def cbt (B, a):
    def ce (cinta, i):
        return i, copiaYBorra(cbt(B, a), B, a)
    return ce

def borraYCopia5 (B, alpha, beta, gamma, delta, epsilon):
    def ce5 (cinta, i):
        return i, cbt ( cbt( cbt ( cbt( cbt (B, epsilon), delta), gamma), beta),alpha)
    return ce5

# cp
def compara (C,U,G,alpha, beta):
    def te_cp2 (C, U, gamma):
        def cp2 (cinta, i):
            if cinta[i] == gamma:
                q = C
            else:
                q = U
            return i, q
        return cp2

    def cp1 (cinta, i):
        cp2 = te_cp2 (C, U, cinta[i])
        return i, buscaMueveIzda (cp2, U, beta)

    return buscaMueveIzda (cp1,buscaPrimero(U,G,beta),alpha)

# cpe (5 argumentos)
def comparaYBorra (C, U, G, alpha, beta):
    def cpe (cinta, i):
        return i, compara ( borra( borra (C,C, beta), C, alpha), U, G, alpha, beta)
    return cpe

# cpe (4 argumentos)
def comparaYBorraTodo (U, G, alpha, beta):
    def cpe (cinta, i):
        return i, comparaYBorra ( cpe, U, G, alpha, beta)
    return cpe

def buscaFin (C):
    def q(cinta,i):
        if cinta[i] != ' ':
            qDevuelto = q
        else:
```

```
            qDevuelto = q1
        i = __R__(cinta, i)
        return i, qDevuelto

    def q1(cinta, i):
        if cinta[i] != ' ':
            i = __R__(cinta, i)
            qDevuelto = q
        else:
            qDevuelto = C
        return i, qDevuelto

    return q

# q
def buscaÚltimo(C,  a):
    def q1bU (cinta, i):
        if cinta[i]==a:
            q = C
        else:
            i = i-1
            q = q1bU
        return i, q

    return buscaFin(q1bU)

# e
def borraMarcas (C):
    def e(cinta, i):
        if cinta[i] =='@':
            i = __R__ (cinta,i)
            q = e1
        else:
            i = i - 1
            q = e
        return i, q

    def e1 (cinta, i):
        if cinta[i] != ' ':
            i = __R__(cinta, i)
            cinta[i]= ' '
            i = __R__(cinta, i)
            q = e1
        else:
            i = __R__(cinta, i)
            q = C
        return i, q
    return buscaÚltimo (e, '@')
```

Tablas Máquina Universal

```
anf = buscaÚltimo (pcon,  ':')
con = config(pfom, 'y')


e = borraTodo(qFinal, 'y')

# fom
def buscaSiguienteInstrucción(cinta, i):
    if cinta[i] == ';':
        i = __R__(cinta, i)
        cinta[i] = 'z'
        i = i - 1
        q = config (comparaConfiguraciones, 'x')
        #q = config(qFinal, 'x')
    elif cinta[i] == 'z':
        i = i-2
        q = buscaSiguienteInstrucción
    else:
        i = i - 1
        q = buscaSiguienteInstrucción
    return i, q

# fmp
def comparaConfiguraciones (cinta, i):
    return i, comparaYBorraTodo(borraTodo(borraTodo(anf, 'x'), 'y'),
    marcaInstrucción, 'x', 'y')

# sim
def marcaInstrucción (cinta, i):
    return i, buscaMueveIzda(sim1, sim1, 'z')
def sim1 (cinta, i):
    return i, config(sim2,' ')
def sim2 (cinta, i):
    if cinta[i] == 'A':
        q = sim3
    else:
        i = i-1
        cinta[i] = 'u'
        i = __R__(cinta, i)
        i = __R__(cinta, i)
        i = __R__(cinta, i)
        q = sim2
    return i, q
def sim3 (cinta, i):
    if cinta[i] == 'A':
        i = i - 1
        cinta[i] = 'y'
        i = __R__(cinta, i)
        i = __R__(cinta, i)
        i = __R__(cinta, i)
```

```
        q = sim3
    else:
        i = i-1
        cinta[i] = 'y'
        q = borraTodo (marcaConfiguraciónCompleta, 'z')
    return i, q

# ml
def marcaConfiguraciónCompleta (cinta, i):
    return i, buscaÚltimo(ml1, ':')
def ml1 (cinta, i):
    if cinta[i] != 'A':
        i = __R__(cinta, i)
        i = __R__(cinta, i)
        q = ml1
    else:
        i = i - 4
        q = ml2
    return i, q
def ml2 (cinta, i):
    if cinta[i]=='C':
        i = __R__(cinta, i)
        cinta[i] = 'x'
        i = i - 3
        q = ml2
    elif cinta[i] == ':':
        q = ml4
    else:
        i = __R__(cinta, i)
        cinta[i] = 'x'
        i = i - 3
        q = ml3
    return i, q
def ml3 (cinta, i):
    if cinta[i] != ':':
        i = __R__(cinta, i)
        cinta[i] = 'v'
        i = i - 3
        q = ml3
    else:
        q = ml4
    return i, q
def ml4 (cinta, i):
    return i, config(izquierda(izquierda(ml5)),' ')
def ml5 (cinta, i):
    if cinta[i] != ' ':
        i = __R__(cinta, i)
        cinta[i] = 'w'
        i = __R__(cinta, i)
        q = ml5
    else:
```

```
        cinta[i] = ':'
        q = muestraResultado
    return i , q

# sh
def muestraResultado (cinta, i):
    return i, buscaPrimero(sh1, inst, 'u' )
def sh1 (cinta,i):
    i = i - 3
    return i,sh2
def sh2 (cinta, i):
    if cinta[i]=='D':
        i = __R__(cinta, i)
        i = __R__(cinta, i)
        i = __R__(cinta, i)
        i = __R__(cinta, i)
        q = sh3
    else:
        q=inst
    return i, q
def sh3 (cinta, i):
    if cinta[i]=='C':
        i = __R__(cinta, i)
        i = __R__(cinta, i)
        q = sh4
    else:
        q=inst
    return i, q
def sh4 (cinta, i):
    if cinta[i] == 'C':
        i = __R__(cinta, i)
        i = __R__(cinta, i)
        q = sh5
    else:
        q = escribeAlFinal( escribeAlFinal (inst, ':'),  '0')
    return i, q
def sh5 (cinta, i):
    if cinta[i] == 'C':
        q = inst
    else:
        q = escribeAlFinal( escribeAlFinal (inst, ':'),  '1')
    return i, q

# inst
def inst (cinta, i):
    return i, buscaÚltimo(izquierda(inst1), 'u')
def inst1 (cinta, i):
    if cinta[i]=='L':
        i = __R__(cinta, i)
        cinta[i]=' '
        q = borraYCopia5(ov, 'v','y','x','u','w')
```

```
    elif cinta[i] ==’N’:
        i = __R__(cinta, i)
        cinta[i]=’ ’
        q = borraYCopia5(ov, ’v’,’x’,’u’,’y’,’w’)
    else:
        i = __R__(cinta, i)
        cinta[i]=’ ’
        q = borraYCopia5(buscaÚltimo(inst2,’A’), ’v’,’x’,’u’,’y’,’w’)
    return i, q
def inst2 (cinta, i):
    i = __R__(cinta, i)
    i = __R__(cinta, i)
    return i, inst3
def inst3 (cinta, i):
    if cinta[i] == ’ ’:
        cinta[i]=’D’
    return i, ov

#ov
def ov (cinta, i):
    return i, borraMarcas(anf)

def marcaConfig(C):
    anf = buscaÚltimo(config(C, ’y’), ’:’)
    return anf


def configInicial (C):
    def b1 (cinta, i):
        i = __R__(cinta, i)
        i = __R__(cinta, i)
        cinta[i] = ’:’
        i = __R__(cinta, i)
        i = __R__(cinta, i)
        cinta[i] = ’D’
        i = __R__(cinta, i)
        i = __R__(cinta, i)
        cinta[i] = ’A’
        i = __R__(cinta, i)
        i = __R__(cinta, i)
        cinta[i] = ’D’
        return i, C

    return buscaPrimero(b1, b1, ’::’)

bSI = buscaSiguienteInstrucción
mC = marcaConfig( bSI )
```