OTROS MODELOS DE CÁLCULO.

La MT no es el único modelo de cálculo, disponemos de:

Máquina de Post.

Dispone de una cinta indefinida dividida en celdas numeradas y un cabezal (o carro). En cada cinta puede haber un "blanco" o una "marca". El carro se mueve a derecha o izquierda de acuerdo con instrucciones específicas y borra o imprime marcas.

En cada momento la cinta tiene un "estado" ("configuración") y el cabezal se encuentra frente a una celda. El estado de la máquina es el estado de la cinta y la posición del carro.

Un programa de Máquina de Post es una secuencia numerada de instrucciones. Estas pueden ser de 5 tipos:

- Movimiento a la derecha-salto de instrucción
- Movimiento a la izquierda-salto de instrucción
- Imprimir marca-salto de instrucción
- Borrar marca-salto de instrucción
- Salto de control.

Lenguaje GO-TO.

Manipulación de variables numéricas.

Considero un lenguaje de programación con variables de entrada Xi, variables locales Zi, Y variable de salida. Las variables tienen enteros positivos.

Conjunto de instrucciones:

V ← V + 1 incrementar 1 el valor de V

V ← V - 1 decrementar 1 el valor de V

IF V≠0 GOTO L

Un programa es una secuencia de instrucciones que actúan sobre las variables. Inicialmente las variables locales y la de salida tienen el valor 0.

En este lenguaje se puede usar Macros(programas resumidos)

Ejemplo: GOTO L

Z ← Z+1

IF Z≠0 GOTO L

Máquinas Unlimited Register Machine.

Una URM está construida por un número ilimitado de registros Ri, cada uno puede contener un entero positivo ri.

El contenido de los registros se altera en respuesta a ciertas instrucciones que puede reconocer. Una lista de instrucciones constituye un programa. Instrucciones posibles:

- Poner a cero un registro Z(n)
- Incrementar el valor del registro en una unidad S(n)
- Transferencia de valor T(m, n)
- Salto de registro J(m, n, q)

Programas de Post-Turing.

Vamos a estudiar el lenguaje de Post-Turing para manipulación de cadenas. En dicho lenguaje se pueden escribir programas que actúan sobre una cinta ilimitada en la que se pueden escribir o leer símbolos. En dicha cinta siempre hay una casilla activa sobre la que se puede escribir un símbolo. Sólo el símbolo que aquí se encuentre es el que se supone observado.

En un paso de cálculo, el símbolo de la casilla se puede leer, ejecutar una instrucción en función del símbolo, escribir un nuevo símbolo y finalmente moverse a la casilla izquierda o derecha.

ESTRUCTURA DE LOS PROGRAMAS:

Un programa es un conjunto de instrucciones para manipular cadenas sobre un alfabeto de trabajo B(incluye símbolo blanco) y uno de entrada A.

Cada instrucción puede tener una etiqueta (palabra de un alfabeto) opcional. Se escriben como [L] al comienzo.

Instrucciones posibles:

- PRINT a: Imprime el símbolo a en la casilla de la cinta donde está el cabezal
- IF a GOTO L: Si el símbolo que vemos en la cinta es a, entonces nos vamos a la instrucción con etiqueta L.
- RIGHT: mueve el cabezal a la derecha
- LEFT: mueve el cabezal a la izquierda
- HALT: termina y acepta

Funcionamiento:

El programa comienza con una cadena u del alfabeto A en una cinta ilimitada y rodeada por blancos, con el cabezal a la izquierda de la palabra. Ejecuta la primera instrucción. Cada vez que termina una instrucción ejecuta la siguiente, excepto si es un salto con IF a GOTO L. Termina si tiene que ejecutar una instrucción que no existe o llega a HALT.

CALCULABILIDAD POST-TURING.

El lenguaje aceptado es el conjunto de cadenas del alfabeto A que comenzando en la configuración inicial llegan a la instrucción HALT.

Sea DcA*. Una función parcial f:D→B* es calculable por un programa Post Turing si para cualquier u∈A* el programa llega a HALT con f(u) en la cinta si u∈D y no termina en caso contrario.

EJEMPLO1:

```
    LEFT PRINT 0
    LEFT PRINT 1
    HALT
```

El programa añade 10 al principio de la palabra de entrada.

EJEMPLO2:

Alfabeto de entrada A = {0, 1}. B ={0, 1, # }.

```
    LEFT
[C] RIGHT
    IF # GOTO E
    IF 0 GOTO A
    IF 1 GOTO C
[A] PRINT #
    IF # GOTO C
[E] HALT
```

Sustituye los ceros de la cadena de entrada por blancos.

Programas con variables-string.

Viene dado por un alfabeto de entrada A, otro de trabajo B. Dispone de una variable X de entrada, un conjunto finito de variables de trabajo Zi y una variable Y de salida.

Instrucciones posibles:

- V ← Av, añade el símbolo a al principio de la variable V.
- V ← V-,elimina el último símbolo de V.
- IF V ENDS a GOTO L, si el último símbolo de la variable V es 'a' → seguir por la instrucción con etiqueta L.
- HALT, termina

Se empieza con X=u (u∈A*), las demás variables con ε. Acepta la palabra si llega a HALT y calcula una función parcial f si llega a HALT con f(u) almacenado en Y cuando f está definida y no para en otro caso.

EJEMPLO: Alfabeto {0, 1} como entrada X = u y calcula Y = u.

```
        IF X ENDS 0 GOTO A
        IF X ENDS 1 GOTO B
        HALT
[A]     X ← X-
        Y← 0Y
        IF X ENDS 0 GOTO A
        IF X ENDS 1 GOTO B
        HALT
[B]     X ← X-
        Y← 1Y
        IF X ENDS 0 GOTO A
        IF X ENDS 1 GOTO B
        HALT
```

MACROS.

Conjunto de instrucciones. Es una forma reducida de escribir programas.

Ejemplo macro1:

IF V ≠ Ɛ GOTO L

Expansión (siendo {a1 , . . . , an } el alfabeto de trabajo):

IF V ENDS a1 GOTO L

IF V ENDS a2 GOTO L

……

IF V ENDS an GOTO L

Ejemplo macro2:

V ←Ɛ

Expansión:

[L] V ← V-

IF V ≠ Ɛ GOTO L

RESUMIENDO INSTRUCCIONES.

Algunos conjuntos de instrucciones se pueden resumir.

Ejemplo:

IF V ENDS a1 GOTO L1

…

IF V ENDS an GOTO Ln

lo resumimos como:

IF V ENDS ai GOTO Li (i=1,. . . ,n)

Equivalencia de modelos de cálculo.

Los siguientes hechos son equivalentes:

- f es parcialmente calculable por una MT
- f es parcialmente calculable por un programa Post-Turing.
- f es parcialmente calculable por un programa con variables.

Los siguientes hechos son equivalentes:

- L es aceptado por una MT
- L es aceptado por un programa Post-Turing.
- L es aceptado por un programa con variables.

Nos basamos en que los tres modelos se pueden simular entre sí.

UN PROGRAMA CON VARIABLES MEDIANTE POST-TURING.

Sean las variables X1,…,Xm Z1,…,Zk y Y. Llamo l=m+k+1. Escribo las variables en el mismo orden como V1,…,Vl. El programa Post-Turing coloca en la cinta el contenido de las variables del siguiente modo: # X1 # …# Xm#Z1#...#Zk#Y#

Al principio de cada instrucción del programa de variables el cabezal estará situado en el blanco justo a la izquierda de X1.

Simularemos cada instrucción del programa de variables como una macro del programa Post-Turing. Necesito las macros:

GOTO L se expande como

IF a1 GOTO L
……
IF an GOTO L

RIGHT (LEFT) TO NEXT BLANK se expande como:

```
[A]     RIGHT (LEFT)
        IF # GOTO E
        GOTO A
```

[**E**] es la etiqueta de la instrucción inmediatamente después de la macro.

MOVE BLOCK RIGHT se expande como:

```
[C]     LEFT
        IF # GOTO A0
        IF ai GOTO Ai (i=1,. . . ,n)
[Ai ]   RIGHT
        PRINT ai
        LEFT
        GOTO C (i=1,. . . ,n)
[A0 ]   RIGHT
        PRINT#
        LEFT
```

ERASE BLOCK se expande como

```
[A]     RIGHT
        IF # GOTO E
        PRINT #
        GOTO A
```

[**E**] es la etiqueta de la instrucción inmediatamente después de la macro.

Si podemos [i] después de una instrucción, será que la repetiremos i veces.

Veamos como simular un programa con variables mediante un programa Post–Turing:

• La instrucción Vj ← ai Vj se simula como:

```
RIGHT TO NEXT BLANK [I]
MOVE BLOCK RIGHT [I–j+1]
RIGHT
PRINT ai
LEFT TO NEXT BLANK [j]
```

• La instrucción Vj ← Vj - se simula como:

```
        RIGHT TO NEXT BLANK [j]
        LEFT
        IF # GOTO C
        MOVE BLOCK RIGHT [j]
        RIGHT
        GOTO E
[C]     LEFT TO NEXT BLANK [j-1]
```

• La instrucción IF Vj ENDS ai GOTO L se simula como:

```
        RIGHT TO NEXT BLANK [j]
        LEFT
        IF ai GOTO C
        GOTO D
[C]     LEFT TO NEXT BLANK [j]
        GOTO L
[D]     RIGHT
        LEFT TO NEXT BLANK [j]
```

• La instrucción HALT se simula borrando la entrada y las variables intermedias mediante ERASE BLOCK[1-1] y parando.

UN PROGRAMA POST-TURING POR UNA MT.

La MT y el programa de Post tendrán los mismos alfabetos de entrada y de trabajo.

La MT tendrá un estado qi por cada instrucción li del programa Post-Turing, más un estado qf (estado final) y otro estado q(k+1) sin transiciones, siendo k el número de instrucciones del programa. El estado inicial es el de la primera instrucción.

Veamos cómo se simulan las instrucciones:

- li=PRINT aj → $\delta(qi , a) = (q(i+1) , aj , S)$, $\forall a \in B$
- li=RIGHT → $\delta(qi , a) = (qi+1 , a, D)$, $\forall a \in B$
- li=LEFT → $\delta(qi , a) = (qi+1 , a, I)$, $\forall a \in B$
- li=IF ak GOTO lj →

  $\delta(qi , ak) = (qj , ak , S)$

  $\delta(qi , a) = (qi+1 , a, S)$, si $a \neq ak$
- li=HALT →

  $\delta(qi , a) = (qf , a, S)$, $\forall a \in B$

UNA MT POR UN PROGRAMA CON VARIBLES.

Iguales alfabetos. Sea B={a1,…,an} incluyendo blanco. El programa con variables dispone de las variables X, Z, Y, teniendo X inicialmente la palabra de entrada. La idea es que en cada momento X contenga lo que hay a la izquierda del cabezal, Z el símbolo que ve el cabezal e Y lo que hay a la derecha.

Macro V ← -V (quito el primer símbolo de V) se expande como:

```
        U ← Ɛ
[A]     IF V ENDS ai GOTO Bi (i=1,. . . ,n)
        GOTO C
[Bi ]   V ← V-
        U← ai U
        GOTO A i=1,. . . ,n
[C]     IF U ENDS ai GOTO Di (i=1,. . . ,n)
        GOTO E
[Di ]   U ← U-
        IF U ≠ Ɛ GOTO Fi
        GOTO C
[Fi ]   V ← ai V
        GOTO C
```

Macro V ← Vaj (añade aj al final de V) se expande como:

```
        U ← Ɛ
        U ← ai U
[A]     IF V ENDS ai GOTO Bi (i=1,. . . ,n)
        GOTO C
[Bi ]   V ← V-
        U← aj U
        GOTO A i=1,. . . ,n
[C]     IF U ENDS ai GOTO Di (i=1,. . . ,n)
        GOTO E
[Di ]   U ← U-
        V← ai V
        GOTO C i=1,. . . ,n
```

• La IF V STARS ai GOTO L tiene la expansión:

```
        U ← Ɛ
[A]     IF V ENDS aj GOTO Bj (j=1,. . . ,n)
        GOTO E
[Bj]    V ← V-
        U← aj U
        IF V ≠ Ɛ GOTO A
        V ← U
        GOTO E (si i ≠ j)
        GOTO L (si i = j) ( j=1,. . . ,n)
```

Inicialmente se ejecutarán las instrucciones:

```
        Y ← Ɛ
        Z ← Ɛ
        Z ← #Z
[A]     IF X ENDS ai GOTO Bi (i=1,. . . ,n)
        GOTO C
[Bi ]   X ← X-
        Z← Ɛ
        Z ← ai Z
        IF X ≠ Ɛ GOTO Di
        GOTO A
[Di ]   Y ← ai Y
        GOTO A (i=1,. . . ,n)
```

Asociaremos a cada estado qi una etiqueta Ai y a cada par (qi,aj) otra etiqueta Bij donde se simulará la transición δ(qi,aj). Todas las transiciones no definidas se pueden asociar a la misma etiqueta E.

En las etiquetas Ai hay las siguientes instrucciones para los estados no finales:

[Ai] IF Z ENDS a1 GOTO Bi1
IF Z ENDS a2 GOTO Bi2
…
IF Z ENDS an GOTO Bin

En las etiquetas Ai correspondientes a un estado final tendremos instrucciones que ponen toda la cinta en Y. Antes de parar habría que quitar los blancos a la derecha y a la izquierda de Y. Si tenemos la transición δ(qi, aj) = (qm, ak , D), entonces ponemos el grupo de instrucciones:

[Bij] X ← Xak
Z ← ε
IF Y STARS al GOTO Cml (l=1,. . . ,n)
Z ← #Z
GOTO Am
[Cml] Y ← -Y
Z ← al Z
GOTO Am (l=1,. . . ,n)

Si tenemos la transición δ(qi,aj) = (qm, ak ,I), entonces ponemos el grupo de instrucciones:

[Bij] Y ← ak Y
Z ← ε
IF X ENDS al GOTO Dml (l=1,. . . ,n)
Z ← #Z
GOTO Am
[Dml] X ← X-
Z← al Z
GOTO Am (l=1,. . . ,n)

Tesis de Church Turing.

Los modelos de cálculo vistos tienen las mismas capacidades para calcular funciones y aceptar lenguajes. La generación de este hecho lleva a:

TESIS DE CHURCH TURING:

Toda función efectivamente calculable (calculable por un proceso mecánico bien definido) puede ser calculada por una MT.

Esta tesis no se puede demostrar matemáticamente si no se da una definición formal de efectivamente calculable.