MÁQUINAS DE TURING:
PROBLEMAS INDECIDIBLES

VERSION 20100922

PRESENTACIÓN

El presente documento surge de los apuntes tomados de diversas fuentes durante mi paso por la Ingeniería Técnica en Informática de Sistemas de la Universidad Nacional de Educación a Distancia.

Tiene pocos ejemplos y ningún ejercicio propuesto: solo es un tratado teórico, poco formal pero riguroso. Este inconveniente se puede solventar ampliamente si se consulta la bibliografía del apéndice A.

Por último, aunque el documento está enfocado a las asignaturas de teoría de autómatas y computación de la UNED, el lector no tendrá problemas en su compresión si tiene conocimientos previos de álgebra elemental, teoría de conjuntos grafos y, complementariamente, autómatas finitos y de pila.

Roberto J. de la Fuente López

INDICE

1.- INTRODUCCIÓN

La primera pregunta que nos vamos a plantear es ¿Qué clase de problemas es capaz de resolver un computador? O, de forma más genérica ¿qué clase de problemas es capaz de computar una máquina? Para contestarla primero tenemos que definir en términos teóricos esta pregunta y su posterior respuesta.

Un problema es una pregunta precisa sobre un conjunto de elementos, que arroja como solución otro conjunto de elementos. De manera informal hemos dado la definición de una función de un conjunto origen (dominio) sobre un conjunto imagen (rango). Si cambiamos la pregunta o el conjunto origen, estamos definiendo otro problema, otra función.

Resolver el problema consiste en aplicar una secuencia finita de acciones no ambiguas sobre los datos de entrada para obtener unos datos de salida, es decir, consiste en aplicar un algoritmo que nos calcule la solución. Se dice que **un problema es computable si el problema definido** (la función) **se puede calcular mediante algún algoritmo**.

Por máquina vamos a considerar cualquier *modelo computacional* de resolución de problemas. Un modelo estará definido por unas primitivas y unas reglas de aplicación de las mismas, que además definirá su capacidad computacional (que tipo de problemas es capaz de resolver). Dos modelos son equivalentes, computacionalmente hablando, si son capaces de resolver los mismos problemas.

Para poder utilizar un modelo computacional, lo primero que hay que hacer es representar simbólicamente los elementos del conjunto origen para que la máquina los entienda. En las máquinas que vamos a considerar siempre vamos a tomar como entrada una cadena de símbolos en el alfabeto de la máquina, la cual pertenece a un lenguaje determinado. Así el conjunto origen lo podemos definir como un lenguaje y cada uno de sus elementos se representa por las cadenas de entrada a la máquina.

El que se haga una u otra representación simbólica de la máquina es indiferente, pues siempre se va a poder convertir de una a otra representación. Lo importante de esta

conversión es que, con la representación en cadenas y lenguajes a los que pertenecen, podemos representar cualquier clase de problema: funciones, grafos, gramáticas, autómatas, máquinas de Turing,….

Los problemas se pueden clasificar según sus características. Una posible clasificación es si un problema es de decisión o no. Un problema es de este tipo si el conjunto imagen (rango) solo tiene dos elementos dicotómicos posibles: {sí, no}, {0,1}, {encendido, apagado}, {alto, bajo}….Se puede demostrar que cualquier problema que no sea de decisión se puede reformular para que así lo sea; es decir, todos los problemas pueden plantearse como de decisión.

Dado que se han codificado los elementos como cadenas que pertenecen a un lenguaje, el problema se puede reformular en términos de **encontrar un algoritmo que sea capaz de *decidir* si una cadena a la entrada de la máquina pertenece o no al lenguaje**.

Para contestar las cuestiones del principio, primero hay que contestar a esta ¿son computables todos los problemas?, es decir, ¿existe al menos un algoritmo para cada uno de los problemas posibles? Como se demostrará más tarde, la respuesta es no.

1.1.- UN POCO DE HISTORIA

Paris 1900: David Hilbert propuso a la comunidad matemática una serie de problemas para resolver en el nuevo siglo que entraba. Inicialmente fueron 10 los problemas expuestos, aunque la lista, finalmente, fue de 23.

1.1.1.- 2º problema de Hilbert

Proponía probar si un sistema axiomático es consistente, es decir, probar si hay un "procedimiento" para determinar la verdad o falsedad de una proposición ("decidir" si es verdadera o falsa"). En realidad por procedimiento se refería a un "algoritmo", concepto que definiremos formalmente mas adelante.

En 1931 Kurth Gödel publicó el famoso teorema de incompletud. Este se enunció para la lógica formal y dice que un sistema axiomático siempre esta incompleto y que no se puede demostrar su validez con esos axiomas que lo definen.

Basándose en los trabajos de Gödel, por un lado Alonzo Church y por otro Alan M. Turing, definieron formalmente el "algoritmo", demostrando que no existía uno capaz de decidir (verdad o falsedad) una proposición matemática en la teoría de números.

Church desarrolló la definición del cálculo lambda (λ-calculus, utilizado en el modelo de programación lógica) y Turing desarrolló la máquina de su mismo nombre. Ambos modelos son computacionalmente equivalentes y concluyeron en la TESIS DE CHURCH-TURING.

Una vez que se definió formalmente el término algoritmo, ya se podía intentar probar el 10º problema propuesto por Hibert. El estudio de este le llevó a Julia Robinson casi toda su carrera (junto con Martin Davis y Hillary Putnam), pero no llegó a encontrar la solución. Fue en 1970 cuando, un joven estudiante Yuri Matijasevic, basándose en los estudios de Julia Robinson, quien la encontró (probó que no existe un método general para resolver el problema propuesto por Hilbert).

1.2.- LA TESIS DE CHURCH-TURING

Como se ha dicho, ambos matemáticos llegaron a la misma conclusión: Church afirmaba que no existía un método de cálculo más avanzado, más poderoso, que el calculo λ y Turing afirmaba que no existía una máquina de computación más potente que la máquina de Turing. Se puede demostrar que las máquinas de Turing y el cálculo-λ son modelos con el mismo poder computacional.

Es una tesis pues nadie ha sido capaz de rebatir esta afirmación y es tan importante porque nos muestra cual es el límite de la computación actual, y por ende, cuales son los límites de los computadores actuales. A partir de esta tesis se desarrolla la teoría de computación y decidibilidad (un problema que una máquina de Turing no es capaz de **decidir**, es un problema que no es computable).

1.3.- ALGORITMO

La definición formal de algoritmo viene dada por la definición formal de la máquina de Turing y de cómo esta realiza los cómputos. Al ser la máquina de Turing el modelo computacional más poderoso que existe (no debemos de olvidar que es conceptual, no real), se puede convertir cualquier computador actual en una máquina de Turing, y por lo tanto se puede describir el comportamiento del algoritmo que en él se ejecuta de una manera formal.

1.4.- CONJUNTOS RECURSIVOS Y CONJUNTOS RECURSIVOS ENUMERABLES

Como sabemos $\mathbb{N}$ es un conjunto infinito contable. Un subconjunto $S \subseteq X$ se dice que es recursivo si existe una función en $\mathbb{N}$ tal que

$f(x)=1$ si $x \in S$ y $f(x)=0$ si $x \notin S$ (en este caso se dice que la función es recursiva primitiva)

es decir, un conjunto es recursivo si tenemos un medio de determinar si un elemento pertenece o no a dicho conjunto. Un ejemplo claro, aplicable directamente a la definición, es que Ø y $\mathbb{N}$ son conjuntos recursivos. Como consecuencia, cualquier conjunto contable (finito o infinito) es **recursivo**[1]. Tanto S como $\overline{S}$ (complementario de S, elementos que no pertenecen a S, teniendo en cuenta aquí que la complementación es una operación cerrada) son recursivos. Observar que Ø es el complementario de $\mathbb{N}$

Un conjunto S es **recursivamente enumerable** si existe una función recursiva por la cual los elementos de S son el conjunto imagen de la función. En otras palabras, un conjunto es recursivamente enumerable si tenemos un medio de enumerar los elementos del mismo.

Como se puede observar, este segundo tipo de conjuntos es menos restrictivo que el recursivo, por lo que se puede afirmar que cualquier conjunto S recursivo, también es recursivamente enumerable. De hecho, para que S sea recursivo, S y $\overline{S}$ han de ser recursivamente enumerables.

[1] No confundir esta recursividad con la recursividad de un lenguaje de programación

Se denominarán *recursivamente enumerable (RE)* a los conjuntos en los que S es RE pero el conjunto complementario $\overline{S}$ de S, no lo es (RE no es cerrado para la complementación).

¿Qué tiene que ver esto con las Máquinas de Turing? Antes se ha dicho que Turing llegó a la conclusión de que no se puede "***decidir***" la verdad o falsedad de una proposición matemática. Esto es lo mismo que decir que el conjunto de las proposiciones matemáticas (basadas en un sistema axiomático) no es recursivo.

Una máquina de Turing sólo es capaz de "***reconocer***" conjuntos recursivos enumerables (luego se verá la diferencia entre decidir y reconocer). En términos de teoría de computación y lenguajes formales, conjunto aquí es equivalente a lenguaje aceptado (conjunto de cadenas que acepta la máquina). Cuando se defina formalmente la Máquina de Turing, retomaremos la capacidad computacional de esta, utilizándola para demostrar todas estas afirmaciones.

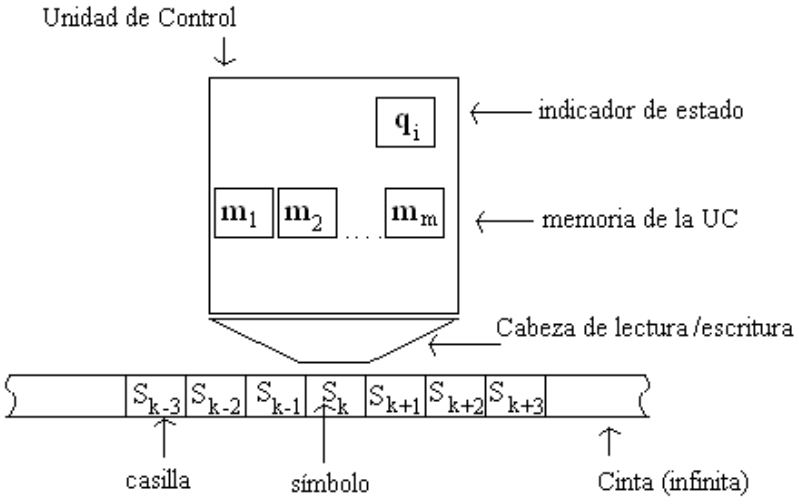
2.- DEFINIENDO LA MÁQUINA DE TURING (MT)

La Máquina de Turing (MT) es conceptual, y se ideó antes de la aparición de los computadores electrónicos. Está basada en cómo un ser humano es capaz de realizar "computaciones". Suponer que se va a realizar un cálculo con lápiz y papel. El papel va a ser el soporte de almacenamiento y el lápiz es el mecanismo que realiza los cambios en aquel. Al principio el papel estará en blanco, donde se irá escribiendo sucesivamente el enunciado de lo que vamos a "computar". En un momento del cálculo, la mente estará en una condición determinada (un estado determinado) de entre un conjunto de ellas (que se puede considerar finito). En esta condición (estado) la atención estará centrada en uno de los símbolos escritos en el papel (este símbolo no tiene porqué ser sólo un carácter). Para avanzar en nuestro cálculo, se hará alguna acción (leer o escribir otros símbolos) basándola en el estado y el símbolo en los que se está centrado, lo que llevará a una condición nueva (un nuevo estado). Cuando se considere que lo que se ha plasmado en el papel es la solución, se deja el lápiz en la mesa, es decir, paramos de calcular. Además, en nuestro

avance también podremos memorizar algún valor intermedio de nuestro cálculo para su posterior uso durante la computación.

Aunque esta máquina fue descrita para realizar cálculos sobre números computables (esto es, números reales cuya parte decimal es calculable en intervalos finitos) de manera que aceptaba la solución cuando esta estaba escrita, se va a definir la Máquina de Turing (MT) en los términos descritos en la introducción: aceptación de una cadena que pertenece a un lenguaje (es otra forma de decir que soluciona un problema).

Así la MT va a estar compuesta por una **unidad de control** (la mente) que va estar en uno de entre un **conjunto finito de estados (q_i)** y que va a tener asociada una **cabeza de lectura/escritura** (el lápiz). Como soporte de almacenamiento va a tener una **cinta** (para hacernos una idea podemos pensar en una cinta de papel), sobre la que apunta la cabeza de lectura/escritura. Esta cinta, que tendrá una longitud infinita por ambos lados (lo que hace que la MT tenga una capacidad infinita de almacenamiento), va a estar dividida en **casillas o celdas** (a una de ellas, y sólo una, apunta la cabeza) y cada una de estas casillas va a contener un **símbolo** (que puede ser más de un carácter)**.** La unidad de control solo puede considerar, en cada paso de cálculo, el símbolo de la casilla a la que apunta la cabeza y el estado en el que se encuentre la unidad de control (UC). Si esta unidad tiene la capacidad de almacenar un conjunto finito de símbolos de entre los que ya se visitaron durante el proceso de computación, entonces tendrá que considerar, además del estado en el que se encuentra, tanto los símbolos almacenados como al que apunta la cabeza (ver variante de MT en 4.2.4).



La figura representa una MT con la unidad de control en el estado q_i, con m celdas de almacenamiento temporal y cuya cabeza apunta a un símbolo S_k en una cinta infinita. Realmente, la forma de representar la MT es indiferente, pues se trata de una máquina conceptual. De momento, vamos a considerar la máquina sin memoria en la UC.

2.1. ¿CÓMO OPERA LA MT?

Primero hay que inicializar la máquina. Al principio, la cinta está en blanco. A continuación se introduce la cadena de entrada en la cinta, que será una secuencia de símbolos del **alfabeto** que entiende, se pone la unidad de control en el estado inicial y se posiciona la cabeza sobre la primera letra de la cadena introducida.

El cambio de condición (de estado), es decir el avance en el cálculo, es lo que vamos a llamar **transición**[2]). Este, como ya hemos dicho, depende del estado actual y del símbolo al que apunta la cabeza. La transición se producirá en el siguiente orden:

- La cabeza lee el símbolo de la casilla a la que está señalando. Teniendo en cuenta este símbolo y el estado en que se encuentra:
    - Cambiamos al siguiente estado. Si resulta que la operación es repetitiva, el estado será el mismo.
    - Escribirá un símbolo en la cinta (los símbolos que se pueden escribir son los del alfabeto y algunos más). Este también puede ser el mismo (esto es equivalente a una operación de solo lectura).
    - **Mover** la cabeza a la izquierda o a la derecha (según se necesite) **una y solo una casilla.**

Durante el cálculo se pueden producir dos situaciones:

- Que se llegue a una solución, en cuyo caso la máquina se parará.

[2] En la bibliografía a esto lo suelen llamar **movimiento**, por la analogía con el movimiento de la cabeza de lectura/escritura. Sin embargo este nombre no es del todo adecuado, pues se puede definir un modelo de MT que permita el no movimiento de la cabeza, efectuando, sin embargo, una transición de la máquina.

- Que el problema no tenga solución, en cuyo caso la máquina puede pararse o entrar en un bucle infinito y no parar nunca. *Este comportamiento dependerá exclusivamente del problema* que se esté intentando computar.

2.2. DEFINICIÓN FORMAL DE LA MT

El modelo de MT que se va a definir no incluye la posibilidad de almacenar un número finito de símbolos en la unidad de control y los símbolos van a estar compuestos por un solo carácter. Además, se utilizará la MT como una máquina que acepta una cadena perteneciente a un lenguaje dado (en el mismo sentido de los autómatas finitos y de pila).

Así una MT se define por una séptupla:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F) \text{ donde:}$$

- Q .- Conjunto finito de estados q_i
- Σ .- Alfabeto de la máquina. Es un conjunto finito de símbolos
- Γ .- Alfabeto de cinta. Es un conjunto finito de símbolos. Por definición el alfabeto de la máquina está incluido en el alfabeto de cinta ($\Sigma \subset \Gamma$)
- δ .- Función de transición. Es un conjunto finito y define el comportamiento de la máquina. Dado un estado q_i y un símbolo de cinta, la MT pasará al estado determinado por la transición, escribirá un símbolo de cinta y se desplazará a la derecha o izquierda según se indique.
- q_0 .- Es el estado inicial de la máquina y es en el que esta empieza a computar. Es decir, $q_0 \in Q$

- B.- Símbolo de cinta que representa la casilla en blanco[3]. Este, que no puede pertenecer al alfabeto de la máquina, puede representarse también con los símbolos Δ , □ o ␣ según convenga. Por definición B pertenece al alfabeto de cinta pero no al de la máquina ($B \in \Gamma / B \notin \Sigma$).

- F.- Subconjunto de Q, y por tanto finito, que contiene los estados finales o de aceptación. Es decir, ($F \subset Q$).

Cuando se inicializa M, las casillas de la cinta están en blanco, estando representadas por el símbolo de casilla en blanco. A continuación se produce la entrada (cadena de símbolos en el alfabeto de la máquina) que se introducirá en la cinta a partir de la posición a la que apunta la Cabeza de lectura/escritura (esta posición es indiferente, pues la cinta es infinita). A partir de aquí se realizan las transiciones que determine la función de transición y la máquina parará cuando, para un estado y un símbolo de cinta, la función de transición no esté definida, aceptando la cadena si el estado es de aceptación.

Se puede decir que una MT siempre parará cuando acepte la cadena. Sin embargo, como ya se ha dicho, *dependiendo del lenguaje* (del problema), no se puede garantizar que la máquina pare cuando la cadena no pertenezca al mismo.

Sea Q' el conjunto de estados de Q exceptuando los de aceptación (Q' = Q – F) y sea S = {I, D} el conjunto de direcciones de movimiento posibles, donde I y D son el movimiento a la Izquierda o a la Derecha respectivamente, la función de transición se define como:

$$\delta: Q' \times \Gamma \rightarrow Q \times \Gamma \times S$$

Por ejemplo, si tenemos la transición $(q_0,a) \rightarrow (q_1,b,I)$, esto significa que si estamos en el estado q_0 y sobre el símbolo de cinta a, procedemos en el orden indicado en el punto 2.1: se pasa al estado q_1, se sustituye el símbolo actual, a, por b, y la unidad de control se mueve una casilla a la izquierda.

[3] En las publicaciones consultadas, a B se le llama espacio en blanco a lo que aquí llamamos símbolo de casilla en blanco. Esta diferenciación la hacemos para distinguir el caso de que el alfabeto contenga el espacio en blanco como tal (separación entre palabras).

¿Porqué necesitamos Q'? Cuando se llega a un estado de aceptación, la máquina debe parar, ya que en un estado de aceptación no se deben hacer más cálculos y por lo tanto no puede ser argumento de la función (no existe una transición desde este estado). Una consecuencia de esto es que el estado inicial no puede ser de aceptación: si así fuera, la máquina, directamente, no haría nada (en este caso acepta el lenguaje vacío). Es decir, para que una máquina de Turing acepte alguna cadena debe tener, como mínimo, dos estados: inicial y de aceptación.

Ejemplo: Sea la máquina de Turing que va a tener 5 estados, numerados de q_0 a q_4; como alfabeto las letras a, b, c y el espacio en blanco ƀ (en el sentido de espacio entre palabras); como símbolos de cinta, los del alfabeto más el de casilla en blanco (que denotamos aquí con Δ, para no confundir con el alfabeto de la máquina), el estado inicial es q_0, y los estados de aceptación son q_2 y q_4. Así la definición formal de la máquina (incompleta, porque nos falta al función de transición) será:

$$M=(\{ q_0, q_1, q_2, q_3, q_4 \}, \{a,b,c,ƀ\}, \{a,b,c,ƀ, \Delta \},$$
$$\{\delta: (Q' \times \Gamma) \rightarrow Q \times \Gamma \times S\}, q_0, \Delta ,\{ q_2, q_4\})$$

--

Se podrá tener una MT que, aunque tenga dos o más estados, ninguno lo sea de aceptación: en este caso la MT no acepta ninguna cadena, con lo que se dice que acepta el lenguaje vacío Ø.

2.3. MUESTRA INFORMAL DEL FUNCIONAMIENTO DE UNA MT

Se mostrará con un ejemplo: Diseñar una MT que compruebe (que se pare) si la cadena de entrada pertenece al siguiente lenguaje

$$L=\{\omega\$\omega / \omega \in \{a + b\}^* \}$$

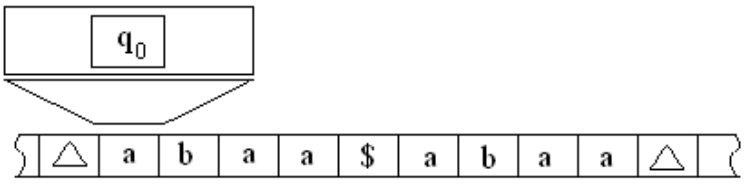
Es decir, comprobar si la cadena de a,s y b,s a la izquierda de $ es igual a la cadena de la derecha. Así podemos definir

$$\Sigma = \{a,b,\$\}$$

$$\Gamma = \{a,b,\$, \Delta ,X\}$$

Hemos utilizado el símbolo Δ para no confundir el símbolo de casilla en blanco con la b del alfabeto de la máquina, y X un símbolo de cinta adicional necesario para la operación.

Inicialización de la máquina: Al principio, la cinta está en blanco. A continuación se introduce la cadena de entrada, por ejemplo abaa$abaa, se pone la unidad de control en el estado inicial y se posiciona la cabeza sobre la primera letra de la cadena introducida.

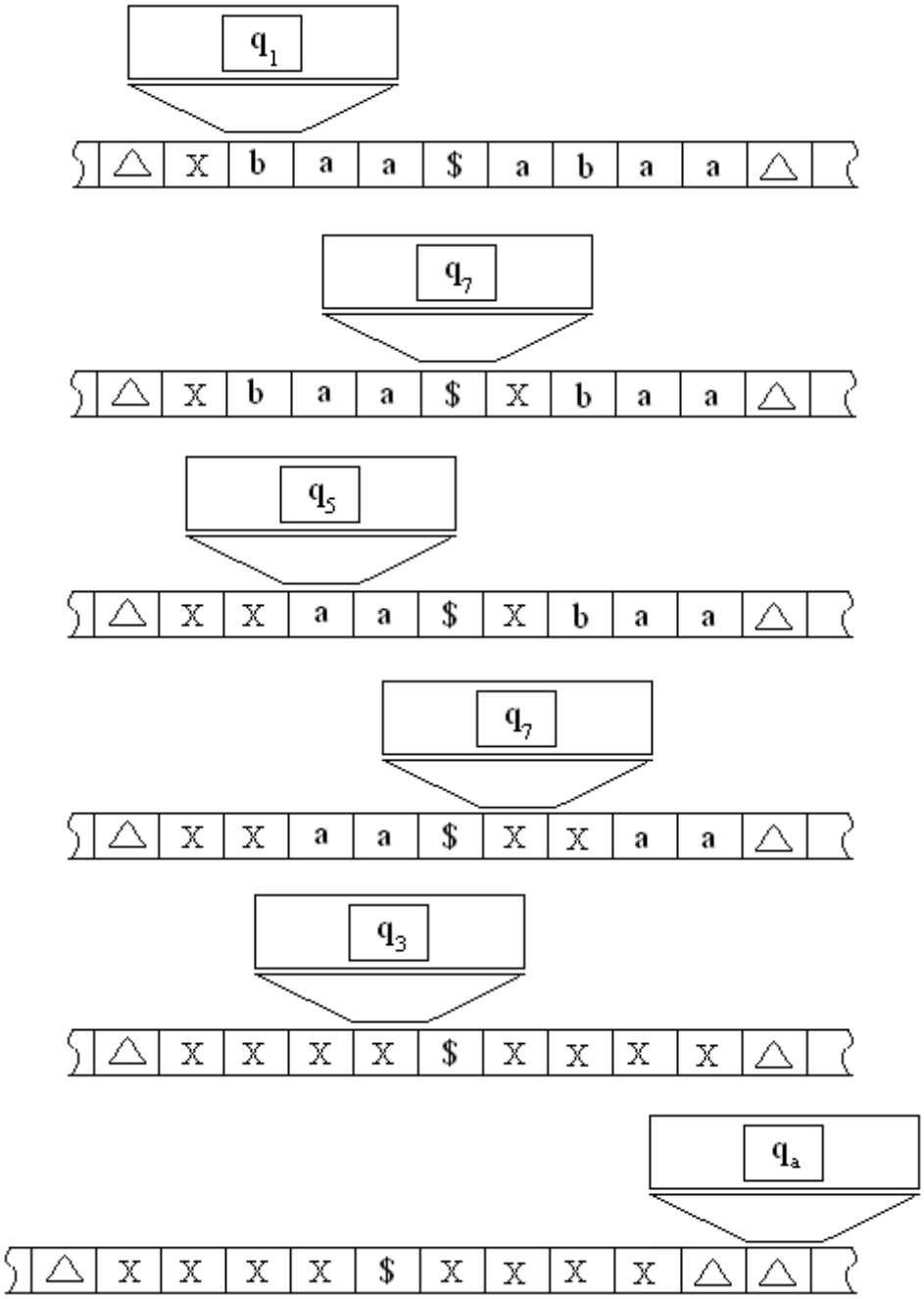


Puesta en marcha de la máquina: Sólo podemos movernos de celda en celda, y lo hará con arreglo a lo que marque la función de transición. La estrategia utilizada en este ejemplo consistirá en realizar un movimiento de ida y vuelta, de manera que vamos comprobando, símbolo a símbolo uno de la subcadena ω a la izquierda de $ con su correspondiente, en la misma posición, de la subcadena ω de la derecha. Esto lo conseguiremos mediante la **técnica del marcado**. En nuestro ejemplo, se marca el primer símbolo (la primera *a* con una X), luego se recorre la cinta hasta la primera letra después de $ y se comprueba si es una *a;* si no lo es la maquina se para en un estado que no es de aceptación, y si lo es, se la marca con una X y se vuelve a la primera letra sin marcar de la subcadena ω de la izquierda. La máquina se parará:

- *Aceptando la cadena*, si se han marcado todas las letras y las subcadenas son iguales, es decir, parándose en un estado de aceptación.

- *Rechazando la cadena* si se produjo una discrepancia entre subcadenas (letras distintas, longitudes distintas), es decir, parándose en cualquier estado que no es de aceptación).

Algunas instantáneas de la máquina a lo largo de la computación son:



El marcado y el movimiento hacia delante y hacia atrás y viceversa, caracterizan el funcionamiento de una MT.

2.4. CONFIGURACIONES

En el ejemplo anterior se han mostrado unas cuantas instantáneas de la MT. A cada una de estas se la llama **configuración**. En esta tenemos representadas tres cosas: el **contenido de la cinta, el símbolo que contiene la casilla** a la que apunta la cabeza de lectura/escritura y **el estado** en el que se encuentra la unidad de control de la maquina en ese instante.

Para no tener que realizar un dibujo de la máquina con cada configuración que se pueden dar en la aceptación/rechazo de una cadena, se va a definir una notación más concisa para representarlas: con cadenas de texto.

El primer problema que se plantea en esta nueva representación es ¿cómo se puede representar el contenido de una cinta si su longitud es infinita?. Como se ha dicho, al principio la cinta está en blanco y a continuación sólo tendrá la cadena de entrada. En este caso los símbolos de casilla en blanco son irrelevantes para representar una configuración, y por lo tanto no se van a representar. Es decir, solo se escribirá desde el primer símbolo de la izquierda de la cadena de entrada hasta el último de esta.

Antes de continuar, es necesario fijar un convenio de notación para las configuraciones:

- Símbolos de entrada.- a, b, c… primeras letras de abecedario en minúscula o una a (minúscula) con subíndices.

- Símbolos de cinta.- Además de B (símbolo de casilla en blanco), Z,Y,X… últimas letras de abecedario en mayúscula o una X (mayúscula) con subíndices.

- Cadena del lenguaje.- z, y, x…. últimas letras de abecedario en minúsculas o una x (minúscula) con subíndice.

- Cadena de símbolos de cinta.- letras griegas minúsculas.

- Estados.- p, q, r, … cuando son pocos. Si son muchos, se suele denotar como q (minúscula) con subíndices (como los mostrados en el ejemplo), empezando

normalmente por el cero. No obstante, y para evitar ambigüedades, hay que tener cuidado con que la notación de los estados no pertenezca al alfabeto de la cinta.

Una configuración se representará de la siguiente manera: la cinta contiene una cadena de símbolos de cinta $\gamma = \alpha\beta$, donde α y β son dos subcadenas que, concatenadas, forman γ. Dado un estado q_i y la cabeza de lectura/escritura posicionada sobre el primer símbolo de la subcadena β, la configuración se representa como

$$\alpha\ q_i\ \beta$$

Así, la primera instantánea del ejemplo anterior en texto estará representada por:

$$X\ q_1\ baa\$abaa$$

Que es mucho más conciso que dibujar la máquina en cada configuración.

Una configuración especial es la **configuración de inicio**: Si ω es la cadena de entrada, y q_0 el estado inicial, la posición de inicio será el primer símbolo de ω. Se representa por

$$q_0\ \omega$$

En nuestro ejemplo, la configuración de inicio estará representada por

$$q_0\ abaa\$abaa$$

Una **configuración de aceptación** es aquella cuyo estado es de aceptación. En el ejemplo, es la última instantánea dibujada.

$$XXXX\$XXXX\ \Delta\ q_a\Delta$$

Una **configuración de rechazo** es aquella cuyo estado es cualquiera que no sea de aceptación y para el que no hay definida una transición.

2.5. TRANSICIONES

Se dice que una configuración C_i **produce** otra C_{i+1} cuando, por aplicación de la función de transición a C_i se obtiene C_{i+1}.

Para una máquina M, vamos a definir el **operador transición** con el símbolo $\vdash_M$, donde el subíndice M es opcional si sabemos cual es la máquina para la que describimos la transición. Podremos denotar cero, una o más transiciones aplicando la estrella de Kleene sobre el operador $\vdash^*_M$

Dado que el modelo de MT que se ha definido es obligatorio mover la cabeza a la derecha o a la izquierda, una producción (transición) será sinónimo de movimiento, el cual se efectuará de forma distinta, dependiendo de si es a la derecha o a la izquierda.

Dados q y p, dos estados para los que existe una transición, X_i es cada uno de los símbolos de cinta en la configuración, Y es el símbolo de cinta que sustituye al que apunta la cabeza de lectura/escritura, X_0, y la UC en el estado q, su configuración es

$$X_{-n}\, X_{-(n-1)} \ldots X_{-2}\, X_{-1}\, q\, X_0\, X_1\, X_2\, X_3 \ldots\, X_{m-1}\, X_m$$

Podremos realizar un movimiento:

- A la izquierda.- Si la función de transición es $(q, X_0) \rightarrow (p, Y, I)$, es decir, se está en el estado q señalando el símbolo X_0, debiendo sustituirlo por Y, pasar al estado p y mover una casilla a la izquierda, la cabeza apuntará al símbolo de cinta anterior al sustituido

$$X_{-n}\, X_{-(n-1)} \ldots X_{-2}\;\; X_{-1}\, \mathbf{q}\;\; \mathbf{X_0}\;\; X_1\, X_2\, X_3 \ldots\, X_{m-1}\, X_m$$
$$\vdash$$
$$X_{-n}\, X_{-(n-1)} \ldots X_{-2}\, \mathbf{p}\;\; \mathbf{X_{-1}}\;\; \mathbf{Y}\;\; X_1\, X_2\, X_3 \ldots\, X_{m-1}\, X_m$$

- A la derecha.- Si la función de transición es $(q, X_o) \rightarrow (p, Y, D)$, es decir, se está en el estado q señalando el símbolo X_0, debiendo sustituirlo por Y, pasar al estado p y

mover una casilla a la derecha, la cabeza apuntará al siguiente símbolo de cinta al sustituido.

$$X_{-n}\, X_{-(n-1)} \ldots X_{-2} \quad X_{-1}\, \underline{\mathbf{q}\ \ X_0}\ \ X_1\, X_2\, X_3 \ldots\, X_{m-1}\, X_m$$
$$\vdash$$
$$X_{-n}\, X_{-(n-1)} \ldots X_{-2} \quad X_{-1}\, \underline{\mathbf{Y\ p}\ \ X_1}\, X_2\, X_3 \ldots\, X_{m-1}\, X_m$$

Tomando la configuración de inicio del ejemplo informal, se va a mostrar el movimiento desde el estado inicial hacia la primera instantánea. Así:

$$q_o\ abaa\$abaa \vdash X\ q_1\ baa\$abaa$$

Y tomando la tercera y la cuarta instantánea, no siendo la segunda una producción de la primera, sino después de varias de ellas:

$$XX\ q_5\ aa\$abaa \vdash^{*} XXaa\$\ q_7\ XXaa$$

Cuando se termine de definir la máquina del ejemplo se verá de forma más clara la representación de los movimientos.

Cada tipo de movimiento (izquierda o derecha) puede dar lugar a dos situaciones especiales en la notación: cuando la cabeza esté en alguno de los extremos de la cadena representable de símbolos de cinta.

- Situaciones para movimiento a la izquierda:

  - Que la cabeza apunte al primer símbolo de cinta que no es casilla en blanco. Habrá que representar la primera casilla en blanco en la configuración (suponer B como símbolo de casilla en blanco). Para la transición $(q, X_n) \rightarrow (p,Y,I)$, con la configuración mostrada, su producción será :

$$\underline{\mathbf{q}\ \mathbf{X_{-n}}}\ X_{-(n-1)} \ldots X_{-2} \quad X_{-1}\ \ X_0\ \ X_1\, X_2\, X_3 \ldots\, X_{m-1}\, X_m$$
$$\vdash$$

$$\underline{\mathbf{p\ B\ Y}}\ X_{-(n-1)}\ldots X_{-2}\ \ X_{-1}\ X_0\ X_1\ X_2\ X_3\ldots.\ X_{m-1}\ X_m$$

- Que la cabeza apunte al último símbolo de cinta que no es casilla en blanco y la transición sea de sustituir el símbolo actual por la casilla en blanco, en cuyo caso esta última la podemos eliminar de la representación, ya que se concatena con el resto de casillas en blanco de la parte derecha de la cinta. Para $(q, X_m) \rightarrow (p,B,I)$:

$$X_{-n}\ X_{-(n-1)}\ldots X_{-2}\ \ X_{-1}\ \ X_0\ \ X_1\ X_2\ X_3\ldots.\ X_{m-1}\ \underline{\mathbf{q\ X_m}}$$
$$\vdash$$
$$X_{-n}\ X_{-(n-1)}\ldots X_{-2}\ \ X_{-1}\ \ X_0\ \ X_1\ X_2\ X_3\underline{\mathbf{\ldots.\ p\ X_{m-1}}}\ \underline{\text{▯}}$$

- Situaciones para movimiento a la derecha

  - Que la cabeza apunte al primer símbolo de cinta que no es casilla en blanco y se tenga que sustituir el símbolo actual por la casilla en blanco, en cuyo caso esta última la podemos eliminar de la representación, ya que se concatena con el resto de casillas en blanco de la parte izquierda de la cinta. Para $(q, X_n) \rightarrow (p,B,D)$:

$$\underline{\mathbf{q\ X_{-n}}}\ X_{-(n-1)}\ldots X_{-2}\ \ X_{-1}\ \ X_0\ \ X_1\ X_2\ X_3\ldots.\ X_{m-1}\ X_m$$
$$\vdash$$
$$\underline{\text{▯}\ \mathbf{p}}\ X_{-(n-1)}\ldots X_{-2}\ \ X_{-1}\ X_0\ X_1\ X_2\ X_3\ldots.\ X_{m-1}\ X_m$$

  - Que la cabeza apunte al último símbolo de cinta que no es casilla en blanco. Hay que incluir en la representación la primera casilla en blanco de la derecha. Para $(q, X_m) \rightarrow (p,B,D)$:

$$X_{-n}\ X_{-(n-1)}\ldots X_{-2}\ \ X_{-1}\ \ X_0\ \ X_1\ X_2\ X_3\ldots.\ X_{m-1}\ \underline{\mathbf{q\ X_m}}$$
$$\vdash$$
$$X_{-n}\ X_{-(n-1)}\ldots X_{-2}\ \ X_{-1}\ \ X_0\ \ X_1\ X_2\ X_3\ldots.X_{m-1}\ \underline{\mathbf{Y\ p\ B}}$$

2.6. REPRESENTANDO LA MT

Cuando se ha definido el modelo de MT, se ha dicho que la función de transición es la que define el comportamiento (produce el conjunto de producciones- transiciones) de la máquina. No se ha dicho que, al ser una función de transición, por cada par (estado, símbolo de cinta) vamos a tener una sola transición (no un conjunto de transiciones posibles), con lo que estamos definiendo una MT *determinista*. Al igual que en los autómatas finitos y de pila, esta definición no es estrictamente cierta, pues se necesitaría tener definidas las transiciones para todas las posibles combinaciones de símbolo de cinta-estado. Este punto se tratará cuando se vean las MT no deterministas.

Por tanto, representar este conjunto "función de transición" es equivalente a representar la MT que describe, que puede ser mediante una **tabla de transiciones** o **mediante un diagrama de transiciones**. Como ocurre en los autómatas finitos y de pila, la representación mediante el diagrama de transiciones es adecuada mientras la máquina sea sencilla, pues en caso contrario puede llegar a ser ilegible.

- Representación tabular.- Tendrá tantas columnas como símbolos de cinta y tantas filas como estados posibles. Una entrada de la tabla podrá:

    - Estar en blanco, en cuyo caso corresponde a una situación con la función de transición sin definir (parada).

    - Contener la terna que describe la producción (transición). Si una transición dada se define por $(q, X_k) \rightarrow (p,Y,I)$, la entrada de la tabla para la columna X_k y la fila q, contendrá la terna (p,Y,I).

- Representación mediante diagrama de transiciones.- Será un pseudo-grafo dirigido donde cada nodo se corresponde con cada estado de la máquina, dibujándose para ello una pequeña circunferencia etiquetada en su interior con el nombre del estado. Uno de ellos será de inicio, estando señalado por una flecha etiquetada con el nombre "INICIO", y uno o más serán de aceptación, dibujándose el nodo con dos circunferencias concéntricas etiquetado en su interior con el nombre del estado.

Cada una de las aristas dirigidas representará cada una de las transiciones definidas para la máquina. Se permiten transiciones sobre el mismo estado o desde un estado hacia otro. Estas aristas estarán etiquetadas con: símbolo de cinta actual, símbolo de cinta que lo sustituye y la dirección del movimiento. Si X es el símbolo actual de cinta, Y el que lo sustituye y S la dirección del movimiento, para la transición (q,X)=(p,Y,S), se podrán encontrar las siguientes notaciones, dependiendo de los autores

$$X/Y\ S \quad X/Y,\ S \quad X;Y,S \qquad X \rightarrow Y,S$$

También se puede simplificar si X e Y son iguales (el símbolo se deja como está)

$$X/S \equiv X/X,\ S \qquad X \rightarrow S \equiv X \rightarrow Y,S$$

O la simplificación cuando hay varias transiciones del mismo tipo entre dos mismos estados.

$$X_1 , X_2 /S \equiv \{ X_1/S , X_2/S \} \qquad X_1 , X_2 \rightarrow S \equiv \{ X_1 \rightarrow S , X_2 \rightarrow S \}$$

Antes se ha definido el conjunto S = {I, D} de las posibles direcciones de movimiento. Pues bien, en la etiqueta de la arista, podremos encontrarnos para Izquierda con I (inicial en español), con L (inicial en inglés) o incluso en modo gráfico con ← y para la Derecha con D (inicial en español), con R (inicial en inglés) o en modo gráfico →.
El símbolo de casilla en blanco, dependiendo del alfabeto de la cinta, se puede representar con B o con cualquiera de los símbolos Δ , □ o ␣.

Ejemplo: definir formalmente y representar la máquina de Turing que acepta cadenas que pertenecen al lenguaje L={ω$ω / ω ∈ {a + b}* }, es decir, comprobar si la cadena de a,s y b,s a la izquierda de $ es igual a la cadena de la derecha.

Como se habrá deducido, se trata de la MT que se introdujo de manera informal, luego ya se tienen definidos los alfabetos de la máquina y de cinta:

$$\Sigma = \{a,b,\$\}$$

y

$$\Gamma = \{a,b,\$, \Delta ,X\}$$

La función de transición se va a representar mediante la siguiente tabla

	a	**b**	**$**	**Δ**	**X**
q_0	q_1,X,D	q_5,X,D	q_8,$,D	-	-
q_1	q_1,a,D	q_1,b,D	q_2,$,D	-	-
q_2	q_7,X,I	-	-	-	q_2,X,D
q_3	q_3,a,I	q_3,b,I	-	-	q_4,X,D
q_4	q_1,X,D	q_5,X,D	q_8,$,D	-	-
q_5	q_5,a,D	q_5,b,D	q_6,$,D	-	-
q_6	-	q_7,X,I	-	-	q_6,X,D
q_7	-	–-	q_3,$,I	-	q_7,X,I
q_8	-	-	-	q_a, Δ,D	q_8,X,D
q_a	-	-	-	-	-

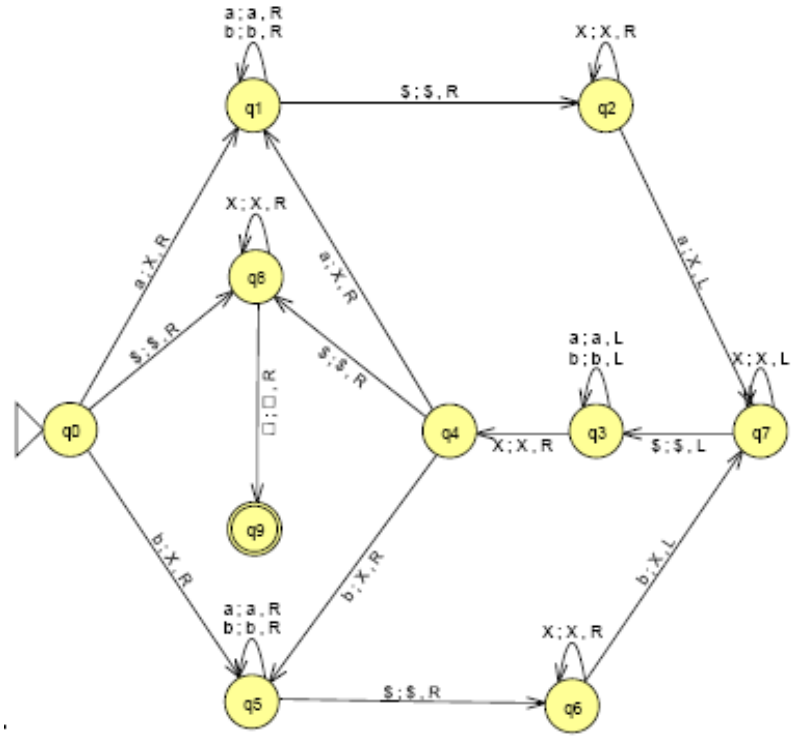
Si la máquina se para en un estado para el que no hay transición pueden ocurrir dos cosas: que la cadena se rechace si se para en uno que no es de aceptación o que la cadena se acepte si lo era de aceptación (de hecho, esta máquina parará siempre).

Así, ya se puede mostrar la definición formal completa[4] de esta máquina:

$$M=(\{ q_0, q_1, q_2, q_3, q_4 ,q_5, q_6, q_7, q_8, q_a \}, \{a, b, \$\} \{a, b, \$, \Delta, X \},$$
$$\{(q_0,a) \rightarrow (q_1,X,D), \ldots , (q_8,X) \rightarrow (q_8,X,D)\}, q_0, \Delta ,\{ q_a\})$$

Al ser esta una máquina sencilla, también se poede representar mediante un diagrama de transiciones (en este diagrama representamos la casilla en blanco con □):

[4] Realmente es una definición formal resumida, pues para que fuera estrictamente completa se deberían haber explicitado todas las transiciones de la máquina.



Analizando este diagrama podemos ver que:

- La rama q_1, q_2 y q_7 comprueba para el símbolo "a" si el símbolo de posición i de ω es igual al símbolo de posición i de la subcadena ω de la derecha.

- La rama q5, q6 y q7 comprueba lo mismo para el símbolo "b".

- La rama q7, q3 y q4 vuelve al principio de la subcadena de la izquierda que queda por comprobar para volver a empezar el proceso con el siguiente símbolo.

- La rama q4 y q8 comprueba que, si no quedan más símbolos de la primera subcadena, tampoco queden más símbolos en la segunda cadena. Si no quedan entonces entra en el estado de aceptación.

Cadenas que acepta: $; abaa $abaa ; a$a ; b$b …
Cadenas que no acepta : $a ; λ[5] ;ab$a ; ab%ab ; ab $ba ; ab$a …

[5] Recordar que λ es la representación de la cadena vacía. Otros autores la denotan como ε.

Ahora se va a simular la secuencia de transiciones para la cadena de entrada abaa$abaa (se han señalado en negrita las configuraciones que corresponden al dibujo de las instantáneas de la introducción informal.)

q_0 abaa$abaa $\vdash$ **X q_1 baa$abaa** $\vdash$ Xb q_1 aa$abaa $\vdash$ Xba q_1 a$abaa $\vdash$
Xbaa q_1 $abaa $\vdash$Xbaa $ q_2 abaa $\vdash$ **Xbaa q_7 $Xabaa** $\vdash$ Xba q_3 a$Xabaa $\vdash$
Xb q_3 aa$Xabaa $\vdash$ X q_3 baa$Xabaa $\vdash$ q_3 Xbaa$Xabaa $\vdash$ X q_4 baa$Xabaa $\vdash$
XX q_5 aa$Xabaa $\vdash$XXa q_5 a$Xbaa $\vdash$ XXaa q_5 $Xbaa $\vdash$ XXaa$ q_6 Xbaa $\vdash$
XXaa$X q_6 baa $\vdash$**XXaa$ q_7 XXaa** $\vdash$ XXaa q_7 $XXaa $\vdash$ XXa q_3 a$XXaa $\vdash$
XX q_3 aa$XXaa $\vdash$X q_3 Xaa$XXaa $\vdash$ XX q_4 aa$XXaa $\vdash$ XXX q_1 a$XXaa $\vdash$
XXXa q_1 $XXaa $\vdash$ XXXa$ q_2 XXaa $\vdash$ XXXa$X q_2 Xaa $\vdash$ XXXa$XX q_2 aa $\vdash$
XXXa$X q_7 XXa $\vdash$ XXXa$ q_7 XXXa $\vdash$ XXXa q_7 $XXXa $\vdash$ XXX q_3 a$XXXa $\vdash$
XX q_3 Xa$XXXa $\vdash$ XXX q_4 a$XXXa $\vdash$ XXXX q_1 $XXXa $\vdash$ XXXX$ q_2 XXXa $\vdash$
XXXX$X q_2 XXa $\vdash$ XXXX$XX q_2 Xa $\vdash$ XXXX$XXX q_2 a $\vdash$
XXXX$XX q_7 XX $\vdash$ XXXX$X q_7 XXX $\vdash$ XXXX$ q_7 XXXX $\vdash$
XXXX q_7 $XXXX $\vdash$ XXX q_3 X$XXXX $\vdash$ XXXX q_4 $XXXX $\vdash$
XXXX$ q_8 XXXX $\vdash$ XXXX$X q_8 XXX $\vdash$ XXXX$XX q_8 XX $\vdash$
XXXX$XXX q_8 X $\vdash$ XXXX$XXXX q_8 Δ $\vdash$ **XXXX$XXXXΔ q_a Δ**

La máquina se parará con una configuración de aceptación (estado q_a), luego la cadena pertenece al lenguaje.

Consideremos ahora la cadena ab$ba, su computación será la secuencia de las siguientes configuraciones:

q_0 ab$ba $\vdash$ X q_1 b$ba $\vdash$ Xb q_1 $ba $\vdash$ Xba$ q_2 ba

La cadena no pertenece al lenguaje porque la máquina se parará en una configuración de rechazo, ya que no hay definida ninguna transición para el estado q_2 y el símbolo b y además q_2 no es de aceptación.

--

A la **secuencia *finita* de configuraciones** que la máquina recorre en la aceptación o rechazo de una cadena, se le llama **historial de cómputo** de la máquina. Si la máquina entra en un bucle infinito no podemos decir que exista un historial de computo (no sería finito).

2.7. DEFINICIÓN FORMAL DE LENGUAJE "ACEPTADO" POR UNA MT

Se acaba de ver como una MT acepta una cadena. Formalmente se dirá que una MT **acepta** la cadena ω si existe una secuencia de configuraciones en la que, si q_0 es el estado de inicio, la primera configuración es $q_0\ \omega$, cada C_{i+1} es una producción de C_i , y la configuración final es de aceptación de la forma $\alpha\ q_a\ \beta$, donde q_a es un estado de aceptación, y α y β son subcadenas de símbolos de cinta. Simbólicamente

$$q_0\,\omega \overset{*}{\vdash_M} \alpha\ q_a\ \beta$$

Dicho de otra forma, una MT, M, acepta la cadena ω si la máquina para en un estado de aceptación. **El conjunto de cadenas aceptadas por la MT es el lenguaje L(M) que acepta esta MT.** Ojo hemos definido un lenguaje aceptado por una MT, no decidido**.** Más adelante se verá esta matización.

3.- VARIACIONES A LA MT DEFINIDA

La descripción mediante MT es un sistema muy robusto, pues admite múltiples variantes sin por ello perder su potencia de computación (aunque tampoco se ha conseguido ampliar). Se van a ver diversas variaciones de MT, las cuales, según la definición dada, serán un modelo computacional distinto. Como la entrada de un modelo computacional se puede codificar para ser la entrada de otro modelo, el problema no cambiará. Se demostrará que las distintas variantes de MT son computacionalmente equivalentes.

Se dice que una MT M simula a otra MT M', si para cada transición de M' hay una secuencia de transiciones de M que realizan la misma operación. De esta manera M realiza lo mismo que M' y quizás alguna operación más, es decir, el lenguaje aceptado por M' está incluido en el lenguaje aceptado por M.

Para demostrar que cualquier variante es equivalente en potencia computacional con la máquina definida originalmente, se procederá de manera que, dadas dos máquinas M y M' primero demostramos que M simula a M' y segundo que M' simula a M. De esta manera demostramos que M y M' aceptan exactamente las mismas cadenas y, por tanto, el mismo lenguaje.

Por la forma en que están construidas las MT, generalmente solo se va a tener que demostrar la simulación en una dirección, pues la otra está implícita en la definición de forma general:

- Si M es una ampliación de M', M aceptará cualquier lenguaje de M'.

- Si M es una restricción de M', M' aceptará cualquier lenguaje que acepte M.

3.1. RETOMANDO LA MT QUE REALIZA CALCULOS

Cuando se definió la MT se hizo una introducción a su concepto y posteriormente se definió como aceptadora de un lenguaje. Hasta ahora lo que se ha hecho es comprobar que la cadena pertenece al lenguaje, y para ello el procedimiento es parar en un estado de aceptación, no importando el contenido final de la cinta. La MT calculadora original utiliza el concepto de *aceptación por parada*: no existe un estado de aceptación: parará siempre cuanto termine el cálculo, pudiendo ser en cualquier estado. El resultado se evalúa por el contenido de la cinta cuando se para: La configuración de la cinta muestra la solución buscada (si existe).

Un detalle importante de esta máquina es que, como es un cálculo, la máquina siempre parará con el resultado en su cinta. Este comportamiento coincide con la definición de algoritmo basado en la MT, ya que un cálculo matemático no deja de ser un procedimiento (algoritmo) ya conocido de realizar algún computo.

La definición formal de esta MT, donde se observa que no hay conjunto de estados de aceptación, será

$$C=(Q, \Sigma, \Gamma, \delta, q_0, B)$$

Para introducir en la máquina los números del cálculo, primero se deben convertir (codificar). Un método muy utilizado es la *notación unaria*[6]: dado un número natural con valor n se codifica con n símbolos consecutivos (n unos o n ceros) y se separa por otro símbolo de marcado (un cero o un uno respectivamente).

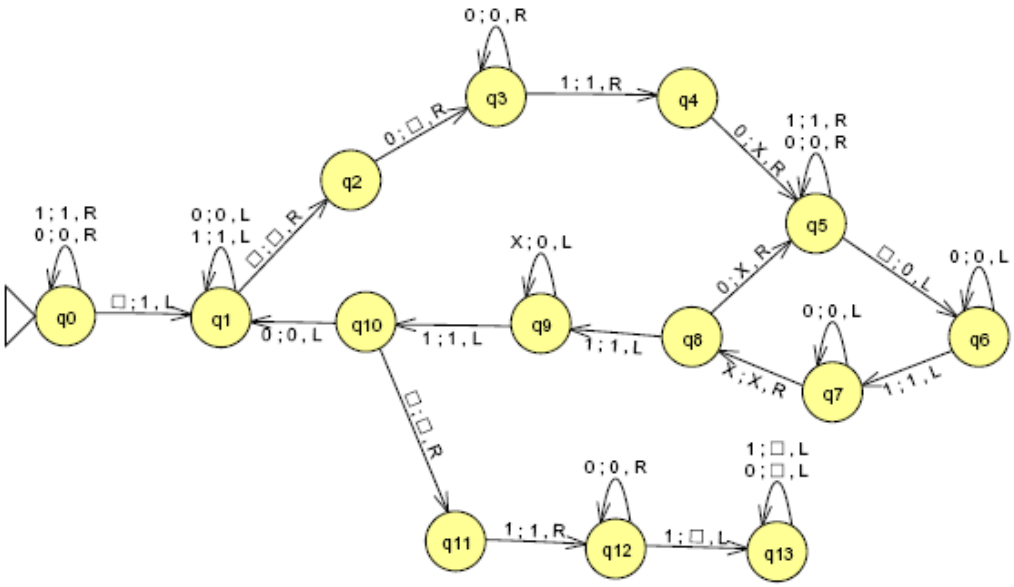
Ejemplo: Construir una máquina que multiplique dos números naturales n y m (mayores que cero). La máquina tendrá una cadena inicial de n ceros seguida de un 1 y seguido de m ceros (0^n 1 0^m), y se parará con la cinta conteniendo *n* veces *m* ceros (0^{nm}). Los componentes de la definición formal de la máquina serán

$Q=\{ q_0 ,\ldots., q_{13}\}$

$\Sigma = \{0,1\}$

$\Gamma = \{0,1,B,X\}$ donde B es la casilla en blanco y X un marcador

$\delta =$



Supongamos que n=2 y m=3, la configuración inicial será:

[6] La notación unaria es históricamente la anterior a la posicional: se corresponde con el concepto de contar por palotes; cuando se llega a un número determinado (por regla general cinco), estos se agrupan (se dibuja una línea de tachado) y se continua añadiendo palotes.

$q_0\,001000$

1- Marcamos el final de la cadena de entrada con un 1, que utilizaremos como marcador de fin de original. A partir de este uno vamos a construir la solución (estados: de q_0 a q_1).

$001000\ q_1 01$

2- Retrocedemos hasta la izquierda de la cadena y eliminamos el primer cero (estados: de q_1 a q_3).

$q_3\,010001$

3- Vamos al primer 1de la cadena. copiamos los m ceros que hay entre los dos unos, al final de la cadena (utilizando la técnica del marcado) (estados: de q_4 a q_5 y m veces el bucle de estados $q_5 - q_6 - q_7 - q_8$).

$01XXX10\ q_6\ 00$

4- Una vez copiado hay que desmarcarlos (estados: q_8, q_9 y q_{10}).

$q_{10}\,010001000$

5- Repetimos los pasos 2, 3 y 4 (m-1) veces más, hasta que el principio de la cadena sea un uno (estados: de q_{10} a q_1 y el bucle de los anteriores).

$q_{10}\ 10001000000$

6- Cuando terminemos las m veces, nos posicionamos en el segundo 1 de la cadena y desde este incluido sustituimos el contenido de la cinta por símbolos de casilla en blanco (estados: q_{10} a q_{13}).

7- La máquina termina con el contenido en la cinta de 0^{nm} .

$1000\ q_{12}\ 1000000 \vdash q_{13}\ BBBBB000000$

--

Para demostrar que esta máquina tiene el mismo poder computacional que el modelo con estado de aceptación, se puede convertir esta máquina calculadora en una máquina aceptadora de cadenas añadiendo, primero, un estado de aceptación y luego relacionando cada estado en el que paraba con la solución en la cinta con el de aceptación, añadiendo, para cada uno de aquellos, tantas transiciones como símbolos de cinta tenga la máquina original. Si el cálculo termina siempre en el mismo estado, se marca este como de aceptación, no necesitándose añadir ninguno más. De esta forma esta máquina simulará el funcionamiento de la máquina calculadora, aceptando cuando se pare con resultado sea positivo y rechazando cuando el resultado sea negativo.

Ahora hay que demostrar lo contrario: una máquina que acepta un lenguaje parándose en un estado de aceptación, con cualquier configuración en la cinta (se dijo que no nos importaba lo que había en ella cuando se paraba), se puede convertir, en una primera fase, en otra MT que acepta el lenguaje con un contenido de cinta determinado (solución), que precisamente es el resultado del cálculo. En una segunda fase se elimina el estado de aceptación, pues con el contenido en la cinta como solución, se da por válido el cálculo (ver variantes de MT, punto 3.2.3 para ver como se procede). Así, la máquina calculadora obtenida aceptará si tiene la solución en la cinta y rechazará si tiene cualquier otro contenido en la misma.

Por tanto, se ha encontrado la forma de simular, en una y otra dirección, las dos máquinas, de manera que se ha demostrado que las dos aceptan el mismo lenguaje. En conclusión, ambas máquinas tienen la misma potencia computacional. A partir de ahora se van a considerar todas las MT como aceptadoras de cadenas.

3.2. OTRAS VARIANTES DE MT

Todas las variantes van a tener en común una capacidad de almacenamiento infinito y se van a diferenciar por una notación distinta para la definición formal y/o la función de transición. Por ejemplo, se puede observar que la definición formal de la máquina calculadora no tenía conjunto de estados de aceptación. Cada una de las variantes que se

mostrarán se podrán utilizar tal cual o construir otras que sean una mezcla de varias de ellas según convenga, habida cuenta de que, con las debidas transformaciones, todas van a ser computacionalmente equivalentes entre sí.

3.2.1. Que solo tenga un estado de aceptación y sólo uno.

La definición formal con esta restricción sería

$$M' = (Q, \Sigma, \Gamma, \delta, q_0, B, q_a)$$

Es fácil convertir la MT original M en una que sólo tenga un estado de aceptación, M', creando un estado de aceptación nuevo, quitando la característica de aceptación a los que antes la tenían y definir, desde cada uno de estos y para cada símbolo de cinta, una transición al nuevo estado de aceptación (para que se pueda mover a este, sea el que sea el contenido de la cinta) sin sustituir el símbolo actual, siendo la dirección del movimiento de la cabeza indiferente. Por tanto M' aceptará los lenguajes que acepte M.

Al ser un caso particular de la MT original en el que F sólo tiene un estado de aceptación, tanto cualquier lenguaje aceptado por M' también lo es por M. Por tanto M y M' aceptan exactamente los mismos lenguajes.

3.2.2. Que tenga un conjunto R de estados de rechazo

Si la máquina para, en lugar de hacerlo en cualquier estado que no sea de aceptación cuando la cadena no pertenezca al lenguaje, esta lo hará en algún estado de rechazo. Así, la MT con esta ampliación se define formalmente:

$$M' = (Q, \Sigma, \Gamma, \delta, q_0, B, F, R)$$

donde R es un conjunto al que pertenecen los estados de rechazo.

Con esta variación lo que realmente se está haciendo es definir completamente una máquina determinista, de manera que, para cada par (estado, símbolo de cinta), hay una transición.

Para convertir la MT original M a esta, M'.se añade a M un conjunto de estados R de rechazo y, para cada entrada de la tabla de transiciones vacía, se añade una transición a cualquiera de los estados de R (la dirección del movimiento de la cabeza es de nuevo irrelevante). M' aceptará o rechazará la cadena si M lo hacía originalmente, luego M' aceptará los lenguajes que acepte M.

Convertir M' en la original, M, bastaría con eliminar los estados de rechazo y las transiciones hacia ellos, de manera que M parará en cualquier estado que no es de aceptación si la máquina con R paraba en un estado de rechazo y continuará aceptando cuando M' acepte la cadena, luego M aceptará los lenguajes que acepte M'. Por tanto, M y M' aceptarán exactamente los mismos lenguajes.

Una variante basada en esta misma, es aquella en la que la máquina tiene sólo un estado de rechazo, donde la máquina se define como

$$M' = (Q, \Sigma, \Gamma, \delta, q_0, B, F, q_r)$$

Es fácil demostrar que esta es equivalente a la anterior utilizando el mismo método que para la variante de un solo estado de aceptación.

3.2.3. Una vez aceptada la cadena, el contenido de la cinta sea un mensaje de aceptación

Esta variante es la que hemos utilizado para convertir una MT que acepta un lenguaje en una MT que realiza un cálculo. Es posible convertir una MT, M, que se para con cualquier contenido en la cinta en otra MT, M', que acepta con un contenido especial, que llamamos "mensaje de aceptación".

Diremos que acepta la cadena con un mensaje de confirmación y la rechaza con cualquier contenido en la cinta. Convertir M en M' es fácil de hacer, sustituyendo el estado de aceptación original (si son varios se convierte primero para que solo tenga uno) por una ampliación de la MT que borre la cinta y escriba un mensaje de aceptación. M' aceptará el lenguaje si lo hacía M, es decir los lenguajes aceptados por M también lo serán por M'.

En el ejemplo del lenguaje L={ω$ω / ω ∈ {a + b}* } con una entrada abaa$abaa, la máquina termina aceptándola, teniendo una configuración en la cinta XXXX$XXXXΔ q_a Δ. Se puede modificar la MT para que borre esta configuración al final y la sustituya por un mensaje de aceptación, por ejemplo ΔYΔ, donde Y es un nuevo símbolo de cinta.

La máquina original es una generalización de esta variación, por lo que cualquier lenguaje que acepta M' también será aceptado por M. Por tanto M y M' aceptan exactamente los mismos lenguajes.

3.2.4. Que se permita no mover la cabeza en una transición

Se permite a la cabeza no moverse cuando se cambia el valor de la casilla actual. En este caso, no es válido utilizar el término movimiento para una transición (se dijo que se podía utilizar como sinónimo en algunos supuestos). Así la máquina queda definida formalmente con

$$M' = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

(igual que la original) y la función de transición cambia su notación por

$$\delta: Q' \times \Gamma \rightarrow Q \times \Gamma \times S'$$

con S'= {I, D, E} donde E es la inicial en español de Estacionario (también se puede ver S de Stationary en inglés)

Otra posible función de transición equivalente podría ser (o se realiza un movimiento con sustitución o se deja estacionaria la cabeza):

$$\delta: Q' \times \Gamma \rightarrow Q \times (\Gamma \cup S)$$

Como se observa en la tabla siguiente, la equivalencia de transiciones entre las funciones de transición mostradas y la de la MT es casi inmediata, por lo que M aceptará los lenguajes que acepte M', y viceversa. Una transición sin movimiento se puede simular, en la máquina original, con una transición adicional que haga una vuelta atrás al símbolo anterior.

Una transición de la máquina original (sustitución y movimiento) se puede simular, en la segunda fila, con dos transiciones y un estado adicional simulando primero la sustitución de símbolo y luego el movimiento.

δ	E	I		D	
δ (Q' x Γ) → Q x Γ x S'	(q,a)→(p,b,E)	(q,a)→(p,a,I)	(q,a)→(p,b,I)	(q,a)→(p,a,D)	(q,a)→(p,a,D)
δ (Q' x Γ) → Q x (Γ u S)	(q,a)→(p,b)	(q,a)→(p,I)	(q,a)→(p',b) (p',b)→(p,I)	(q,a)→(p,D)	(q,a)→(p',b) (p',b)→(p,D)
δ (Q' x Γ) → Q x Γ x S	(q,a)→(p,b,D) (p,b)→(p,b,I)	(q,a)→(p,a,I)	(q,a)→(p,b,I)	(q,a)→(p,a,D)	(q,a)→(p,a,D)

3.2.4. Que la Unidad de control tenga una memoria finita de símbolos

En el punto 2 se ha definido la MT con la unidad de control en el estado q_i, y con un número finito m de celdas de almacenamiento temporal. Estas celdas podrán memorizar algún símbolo de cinta en algún momento de la computación para utilizarla en otro momento de la misma.

Antes de definir formalmente la MT que simula esta unidad de control, primero se debe redefinir el conjunto de estados. Si M_i representa cada una de las celdas de almacenamiento, se define un estado como

$$Q_m = \{Q \times M_1 \times M_2 \times M_3 \times \ldots \times M_m\}$$

donde cada M_i podrá contener cualquier símbolo de cinta de la máquina. Es decir, cada estado ahora será una tupla del producto cartesiano de cada estado original por el alfabeto de cinta para cada una de las celdas de memoria. Así, se define la máquina M' como

$$M' = (Q_m, \Sigma, \Gamma, \delta, (q_0, B_1, B_2, B_3, \ldots B_m), B, F_m)$$

donde $F_m \subset Q_m$ y cada B_i en el estado inicial (para $0 < i \leq m$), es el contenido de cada memoria, que en el estado inicial será el símbolo de casilla en blanco.

La función de transición ahora será

$$\delta: Q'_m \times \Gamma \rightarrow Q_m \times \Gamma \times S$$

donde $Q'_m = Q_m - F_m$ y en cada transición se podrá cambiar, o no, el contenido de alguna o todas las memorias.

Para mostrar el funcionamiento de esta máquina, se va a utilizar el ejemplo informal, el cual se va a modificar para que tenga una celda de memoria en su unidad de control. Si

$$L = \{\omega\$\omega \,/\, \omega \in \{a+b\}^*\}$$

$$\text{donde} \quad \Sigma = \{a,b,\$\}$$

$$\text{y} \quad \text{donde } \Gamma = \{a,b,\$, \Delta, X\}$$

Se va a utilizar la memoria para identificar si el elemento i-ésimo de la subcadena ω de la izquierda es igual al elemento i-ésimo de la subcadena ω de la derecha. Los símbolos de cinta para la memoria serán {a,b, Δ}. La tabla de transiciones será la siguiente (observar que no se han dibujado todas las filas correspondientes a Q_m, ya que algunas no tienen trasiciones definidas):

	a	b	$	Δ	X
q_0, Δ	[q_1,a],X,D	[q_1,b],X,D	[q_8,Δ],$,D	-	-
q_1,a	[q_1,a],a,D	[q_1,a],b,D	[q_2,a],$,D	-	-
q_1,b	[q_1,b],a,D	[q_1,b],b,D	[q_2,b],$,D	-	-
q_2,a	[q_7,a],X,I	-	-	-	[q_2,a],X,D
q_2,b	-	[q_7,b],X,I	-	-	[q_2,b],X,D
q_7,a	-	-	[q_3,a],$,I	-	[q_7,a],X,I
q_7,b	-	-	[q_3,b],$,I	-	[q_7,b],X,I
q_3,a	[q_3,a],a,I	[q_3,a],b,I	-	-	[q_4,a],X,D
q_3,b	[q_3,b],a,I	[q_3,b],b,I	-	-	[q_4,b],X,D
q_4,a	[q_1,a],X,D	[q_1,b],X,D	[q_8, Δ],$,D	-	-
q_4,b	[q_1,a],X,D	[q_1,b],X,D	[q_8, Δ],$,D		
q_8, Δ				[q_a, Δ], Δ,D	[q_8, Δ],X,D
q_a, Δ	-	-	-	-	-

Las transiciones se pueden representar de una manera más concisa. Si Z e Y son la representación de un símbolo de cinta cualquiera, pudiendo ser en este caso *a* o *b* indistintamente, se pueden agrupar las transiciones. Por ejemplo, las transiciones [q_0, Δ], a = [q_1, a],X,D y [q_0, Δ], b = [q_1, b],X,D tienen la misma estructura, diferenciándose solo por el símbolo en la cinta. Si se sustituye por Z, se podrán agrupar las dos transiciones en una: [q_0, Δ], Z = [q_1, Z],X,D.

Para la tabla anterior las transiciones agrupadas podrán ser (se marca en negrilla las transiciones que realmente utilizan la memoria):

[q_0, Δ], Z = [q_1, Z],X,D [q_0, Δ], Z = [q_8,Δ],$,D [$q_1$,Z],Y = [$q_1$,Z],Y,D

[q_1,Z], $ = [q_2, Z],$,D **[$q_2$, Z],X = [$q_2$, Z],X,D** **[$q_2$, Z],Z = [$q_7$,Z],X,I**

[q_7,Z], $ = [q_3, Z],$,I [$q_7$,Z], X = [$q_7$, Z],X,I [$q_3$,Z],Y = [$q_3$,Z],Y,I

[q_3,Z], X = [q_4, Z],X,D [q_4, Z],Y = [[q_1, Y],X,D [q_4, Z], $ = [q_8, Δ],$,D

[q_8, Δ], Δ = [q_a, Δ], Δ,D [q_8, Δ], X = [q_8, Δ], X,D

Para convertir esta máquina M' con transiciones agrupadas en una que no tenga memoria, M, procedemos de la siguiente forma:

- Se expanden las representaciones de símbolos de cinta agrupados (en nuestro caso Z e Y). El resultado se plasma en una tabla de transiciones (como la que se ha mostrado en el ejemplo).
- Para cada estado de Q_m, se le asigna un nombre de estado de la máquina sin memoria y se sustituye esta asignación en la tabla. El resultado es una tabla de transiciones sin memoria.
- Se comprueban las filas de la tabla para eliminar aquellos estados que tengan las mismas transiciones.

Toda cadena aceptada por M' será aceptada por M, luego M aceptará los lenguajes que acepte M'. La máquina original M es un caso particular de M' sin memoria, luego M' aceptará cualquier lenguaje que acepte M. Por tanto, M y M' aceptan exactamente los mismos lenguajes.

Continuando con el ejemplo, si se aplican estos pasos, el resultado es la tabla de transiciones de la máquina original (no tiene porqué ser así).

La representación con el diagrama de transiciones será igual que sin memoria, con la salvedad de que los estados estarán etiquetados ahora por la tupla estado (q_i , Z_1, Z_2, …. Z_m) donde cada Z_i será el contenido de la memoria i.

3.2.5 Cinta con varias pistas

Con anterioridad se dijo que un "símbolo" de cinta no tiene porqué ser un solo carácter; pues bien, este es el caso. Se divide la cinta en **bandas paralelas**, que se van a llamar **pistas**, y, a su vez, se van a dividir en casillas tal como se hizo con la cinta única, resultando en una matriz con tantas filas como pistas y con un número infinito de columnas a ambos lados de la cabeza.

Hay que tener en cuenta que no es lo mismo un símbolo de cinta y un símbolo de pista. Ahora un símbolo de cinta va a ser una tupla de i símbolos de pista, donde i es el número de pistas. Para ello, la máquina accederá solidariamente al contenido de todas las casillas de la misma columna, es decir, accederá a la i-tupla de símbolos de pista.

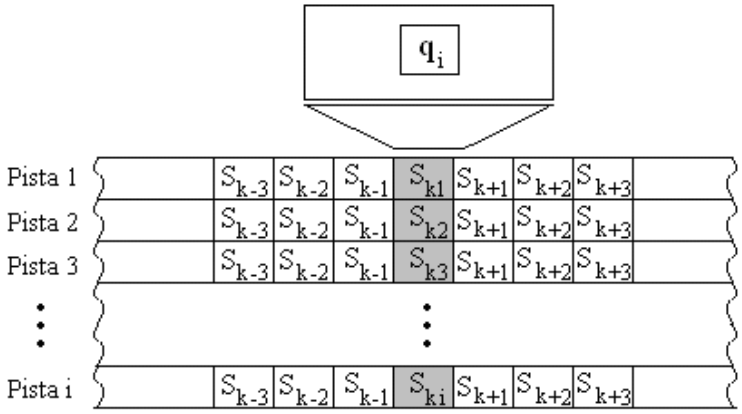
El conjunto de símbolos, para cada pista, podrá ser el mismo o distinto y el conjunto de símbolos de cinta de la máquina serán las tuplas pertenecientes al producto cartesiano de los conjuntos de símbolos de todas las pistas. Si se tienen i pistas, y cada Γ'_k (para $1 \leq k \leq i$) es el conjunto de símbolos de la pista k, el alfabeto de cinta de M' será el conjunto de i-tuplas

$$\Gamma' = \{\Gamma'_1 \times \Gamma'_2 \times \Gamma'_3 \times \ldots \times \Gamma'_i\}$$

Y un símbolo de cinta de M' será, si estamos en la columna k, una i-tupla de símbolos atómicos de la forma

$$(S_{k1}, S_{k2}, S_{k3}, \ldots S_{ki}) \in \Gamma'$$

Una posible presentación de la MT de varias pistas podría ser



Una forma típica de utilizar una máquina de este tipo es colocar la cadena de entrada en la primera pista y utilizar el resto para marcar lo que se necesite (que se ha visitado un símbolo, recordar una posición, dar un mensaje de aceptación….). Al principio, todas las pistas están en blanco, y a continuación se introduce la cadena de entrada. En este instante todas las pistas, a excepción de la primera, están vacías. Así cada símbolo del alfabeto de la máquina será una i-tupla del conjunto

$$\Sigma' = \{\Sigma \times B_2 \times B_3 \times \ldots \times B_i\}$$

donde Σ es el alfabeto de la cadena de entrada en símbolos atómicos. La definición formal de la máquina de varias pistas será

$$M' = (Q, \Sigma', \Gamma', \delta, q_0, (B_1,B_2,B_3,\ldots B_i), F)$$

donde (B_1, B_2, B_3 ,…., B_i) es el símbolo de casilla en blanco. La función de transición sería

$$\delta: Q' \times \Gamma' \rightarrow Q \times \Gamma' \times S$$

que, desglosada en las partes descritas, queda como

$$\delta: Q' \times (\Gamma'_1 \times \Gamma'_2 \times \Gamma'_3 \times \ldots \times \Gamma'_i) \rightarrow Q \times (\Gamma'_1 \times \Gamma'_2 \times \Gamma'_3 \times \ldots \times \Gamma'_i) \times S$$

Por ejemplo:

$$(p, (Z_1,Z_2,Z_3,..,Z_k)) \rightarrow \quad (q , (Y_1, Y_2, Y_3, .., Y_k), I)$$

Significa que una MT en un estado p y símbolo de cinta (Z_1,Z_2,Z_3,..,Z_k), realizará un movimiento al estado q, escribiendo en la cinta el símbolo (Y_1, Y_2, Y_3, .., Y_k) y moviendo su cabeza a la Izquierda.

Para simular una máquina de Turing M' de varias pistas con una M de una sola pista con símbolos de cinta atómicos, para M se definirá:

- Si $\Sigma' = \{\Sigma \times B_2 \times B_3 \times \ldots \times B_i\}$, Σ es el alfabeto de máquina de M
- $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \ldots \cup \Gamma_i \cup \{\#,@\}$. Estas últimas son símbolos de marca adicionales.
- La unidad de control va a tener i celdas de memoria. Esto es válido dado que el número de pistas es finito.

Primero se concatena el contenido de cada una de las pistas en la cinta de M, separadas por el delimitador #. El principio de la primera cadena y el final de la última también se

delimitan con #. Para denotar la posición de la cabeza en la cinta de M', se introducirá la como marca @ y se hará en cada uno de los contenidos de las pistas introducidas en la cinta de M'. Si ω está en la tupla de entrada $(\omega,B_2,B_3,\ldots.B_i)$ para M', la cinta de M quedará con (i+1) delimitadores e i cadenas entre ellos, cada una marcada con la posición de la cabeza. En el estado inicial la cinta de M será:

$$\#@\omega\#@\ B_2\#....\#\ @\ B_i\ \#$$

Para cada transición de M' se simulará la siguiente secuencia de transiciones en M:

- M recorrerá la cinta hasta el último delimitador #, procediendo en su recorrido a almacenar en la celda de memoria M_k (para $1 \leq k \leq i$) el símbolo actual de la cinta k (elemento k de la i-tupla de M'). De esta forma la unidad de control tiene acceso al símbolo de cinta actual de M' (i-tupla).

- Aplicará la transición de M'. M recorrerá de nuevo la cinta y en cada subcadena que representa cada pista, en la posición de la marca de cabeza, se escribe, dependiendo del símbolo en la memoria y el estado, el símbolo que indica la transición, y realiza el movimiento que determina S. Utilizará la memoria para almacenar el símbolo de pista de M' después de la transición.

- Si en una subcadena que representa una pista, después del movimiento la cabeza, esta apuntara al delimitador, a partir de este (inclusive) hay que desplazar una casilla a la derecha el resto de subcadena, y añadir un símbolo de casilla vacía en el hueco que hemos hecho. Por ejemplo, se ha marcado en negrilla la situación y subrayada la subcadena desplazada a la derecha

$$\#\ \alpha\ @\ \beta\ \mathbf{\#@\ Bq_x}\#....\#\ @\ B\ \#\ \vdash\ \#\ \alpha\ @\ \beta\ \mathbf{\#@\ Bq_xB}\underline{\#....\#\ @\ B\ \#}$$

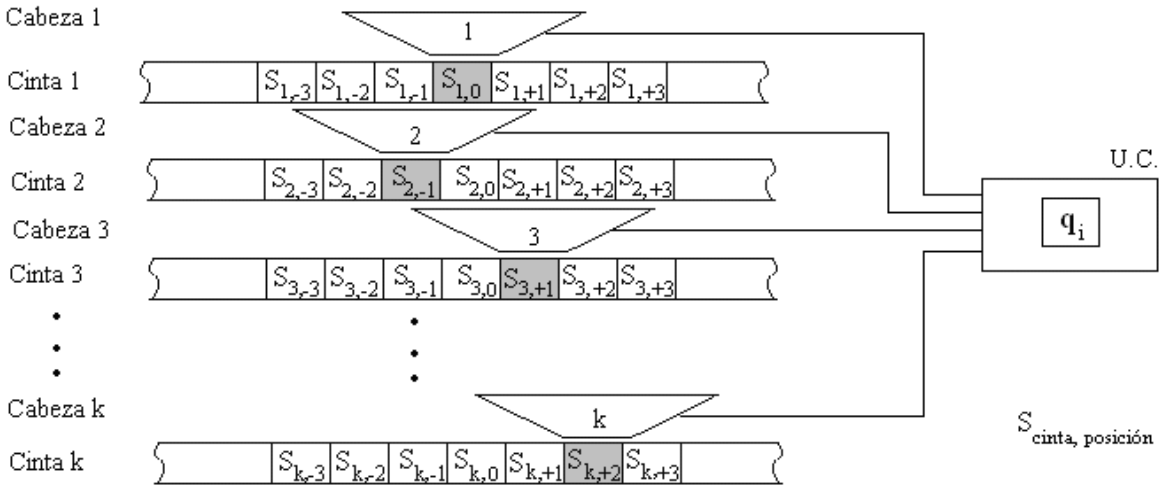
M acepta la cadena en el mismo estado en el que lo haría M', luego M aceptará los mismos lenguajes que aceptaría M'.

Se dijo con anterioridad que la codificación de la entrada a una MT es indiferente, ya que se puede convertir con otra MT. La ampliación de la cinta a varias pistas no deja de ser eso mismo: una codificación distinta para una misma entrada y por tanto esta MT tampoco

ofrece una mejora con respecto a la MT original: la máquina de varias pistas es una generalización de la máquina de una pista (tuplas de un solo elemento) y por tanto cualquier lenguaje aceptado por la máquina de pista única (ya se vio que la MT con memoria es equivalente a la MT sin memoria), lo será por una máquina de una cinta con varias pistas. Por tanto, aceptan exactamente los mismos lenguajes.

3.2.6. MT con varias cintas

Esta variante, que es una generalización de la MT con una cinta, es importante porque *con ella se simula de manera directa el comportamiento de un computador*. El hecho de que la MT de una cinta sea un caso particular de la de varias cintas hace que el lenguaje aceptado por la primera también sea aceptado por la segunda. Después de definir esta máquina se procederá a demostrar que existe una MT de una cinta que simula el funcionamiento de una máquina con varias cintas, y por tanto cualquier lenguaje aceptado por una MT de una cinta también lo será por la MT de varias cintas. Esto demuestra que una máquina con una cinta acepta exactamente los mismo lenguajes que una máquina con varias cintas.



Esta MT estará compuesta por varias cintas como soporte de almacenamiento y una unidad de control que estará en uno de entre un conjunto finito de estados. Tendrá asociadas tantas cabezas de lectura/escritura como cintas, independientes todas ellas entre sí. En cada paso del cálculo la **unidad de control** va a tener en cuenta **el estado** en el que se encuentra y **el contenido de la casilla a la que apunta cada una de las cabezas.**

En el estado inicial, la primera cinta contendrá la cadena de entrada, codificada en el alfabeto de entrada, y el resto de cintas estarán en blanco (se podrán utilizar durante la computación para comprobar que se ha visitado un símbolo, recordar una posición, dar un mensaje de aceptación….). La cabeza de la cinta 1 estará apuntando al primer símbolo de la entrada y la cabeza en el resto de cintas podrá apuntar a cualquier casilla, dado que aquellas siguen en blanco.

Se define un único alfabeto de cinta para la máquina, de manera que cualquier cinta podrán contener cualquier símbolo de este. A este alfabeto de cinta, pertenece el alfabeto de la máquina (en el que estará codificada la entrada en la primera cinta). Podríamos haber definido un alfabeto de cinta para cada una de ellas, pero esto, además de innecesario, solo complicaría la máquina, ya que el alfabeto de cinta para la máquina se podría obtener por la unión de todos los alfabetos independientes de cinta (como se hizo en la definición de MT de varias pistas).

La MT de varias cintas se define formalmente como

$$M= (Q, \Sigma, \Gamma, \delta', q_0, B, F)$$

Se ha dicho que cada cabeza es independiente; esto nos lleva a que, para cada cinta, haya una función de transición de la forma

$$\delta: Q' \times \Gamma \rightarrow Q \times \Gamma \times S'$$

(es la misma que hemos definido para la máquina de una cinta, con la excepción de S', definida anteriormente con la transición estacionaria[7]). Si la máquina tiene k cintas, se podrá generalizar la función de transición a

$$\delta': Q' \times \Gamma^k \rightarrow Q \times \Gamma^k \times S'^k$$

[7]Una MT con transición estacionaria se puede simular con una MT si transición estacionaria. Se define así la máquina para que esta sea más simple.

Así, la función de transición en una MT con k cintas pertenecerá al producto cartesiano de (la notación subíndice se debe identificar como número de cinta y no como un alfabeto de cinta distinto):

$$\delta': Q \times \Gamma_1 \times \Gamma_2 \times \Gamma_3 \times \ldots \times \Gamma_k \rightarrow Q \times \Gamma_1 \times \Gamma_2 \times \Gamma_3 \times \ldots \times \Gamma_k \times S'_1 \times S'_2 \times S'_3 \times \ldots \times S'_k$$

Por ejemplo, la transición:

$$(p, Z_1, Z_2, Z_3, .., Z_k) \rightarrow (q, Y_1, Y_2, Y_3, .., Y_k, I_1, D_2, D_3, .., E_k)$$

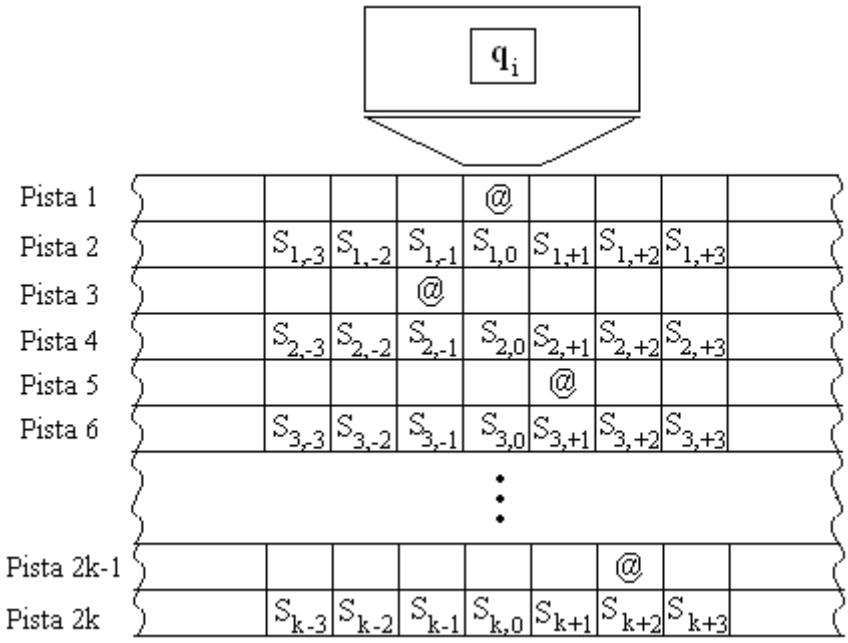
significa que una MT con k cintas que esté en un estado p y con cada cabeza i de lectura/escritura apuntando al símbolo Z_i de cinta, realizará un movimiento a un estado q, sustituyendo cada Z_i por el Y_i que corresponda y realizando el movimiento que corresponda a cada cabeza i.

Ya se ha mencionado que esta máquina va a tener la misma potencia que la de una cinta con pista única. Para demostrarlo, se utilizará un paso intermedio: simular esta MT con k cintas, M, con una MT, M', de una cinta dividida en 2k pistas y que además va a tener k+1 memorias (antes ya se demostró que la MT de varias pistas es equivalente a una MT de una cinta con pista unitaria, y que la MT sin memoria es equivalente a la MT con ella).

Primero se visualizan las k cintas en una cinta de 2k pistas, de manera que la pista 2i (para $1 \leq i \leq k$) es el contenido de la cinta i y la pista 2i-1 es un marcador a la posición actual de la cabeza (pistas impares marcadores, pistas pares contenido de la cinta).

El alfabeto de pista para las impares será {B,@} y para las impares será el mismo que para las pistas de M, Γ. El alfabeto de cinta de M' será un conjunto de tuplas del producto cartesiano de la forma (la notación subíndice se debe identificar como número de cinta y no como un alfabeto de cinta distinto)

$$\Gamma' = \{ \{B,@\} \times \Gamma_1 \times \{B,@\} \times \Gamma_2 \times \{B,@\} \times \Gamma_3 \times \ldots \times \{B,@\} \times \Gamma_k \}$$



En el dibujo de M' con 2k pistas, el símbolo de cinta actual será la tupla

$$(@, S_{(1,0)}, B, S_{(2,0)}, B, S_{(3,0)}, \ldots, B, S_{(k,0)})$$

Para cada movimiento de M, M' tendrá las siguientes transiciones:

- La cabeza de M' buscará consecutivamente a los k marcadores de posición de las k cintas, y almacenará en cada memoria de la UC el contenido de la casilla que marca. Para saber en qué cinta está (pista 2i = cinta i) utiliza la última celda de memoria de la UC. Una vez hecho esto, tenemos el estado y el símbolo actual de cada cabeza de M almacenado en M'.

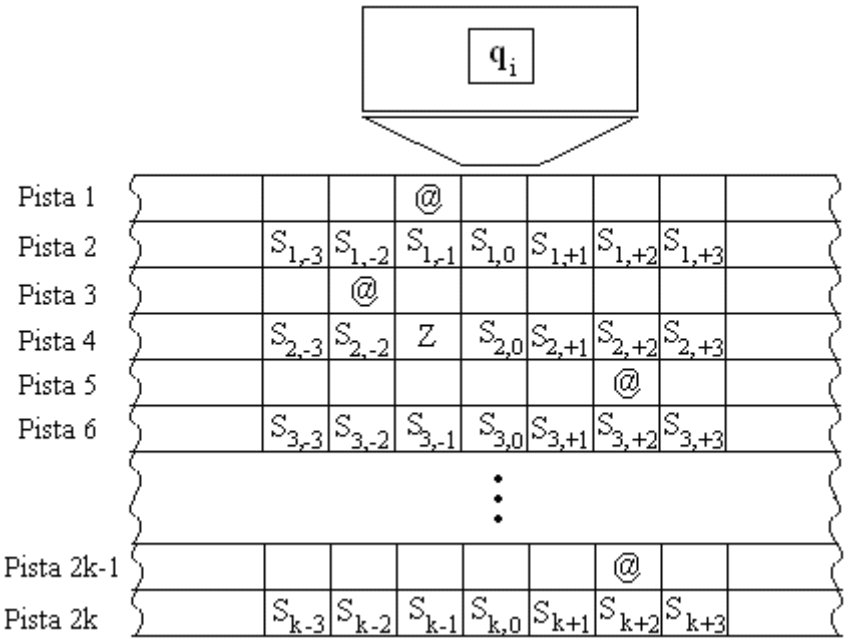
- Para las pistas correspondientes a cada cinta i, M' vuelve a leer el contenido del marcador de la pista 2i-1 y la casilla correspondiente de la pista 2i: Escribe el símbolo que determina la transición correspondiente a la cinta i y mueve la marca (izquierda, derecha o estacionaria) a la casilla que determine la transición. Al final de las 2k pistas, M' tendrá varias tuplas que representan el contenido de las k cintas y la posición de cada cabeza sobre ellas después del movimiento. Utilizará la memoria para almacenar el símbolo de cada cinta de M después de cada transición.

- Los estados de aceptación de M' serán los estados de aceptación de M, luego cualquier lenguaje aceptado por M' también lo será por M.

En el ejemplo del dibujo, dada la transición de M:

$$(p, S_{(1,0)}, S_{(2,-1)}, S_{(3,1)}, \dots ,S_{(k,+2)}) \rightarrow (q, S_{(1,0)}, Z, S_{(3,1)}, \dots ,S_{(k,+2)} ,I,I,D,\dots,E)$$

- M' buscará primero la tupla $S_{(1,0)}$, (@, $S_{(1,0)}$, B, $S_{(2,0)}$, B, $S_{(3,0)}$, … ,B , $S_{(k,0)}$), almacenando $S_{(1,0)}$, en la memoria 1, luego la tupla correspondiente a la segunda cinta (B, $S_{(1,-1)}$, @ , $S_{(2,-1)}$, B, $S_{(3,-1},\dots,B,S_{k-1})$, almacenando $S_{(2-1)}$ en la memoria 2 y así sucesivamente hasta tener en las k primeras memorias los símbolos actuales de M.
- La transición que se está computando modifica el contenido de la segunda cinta, y mueve cada uno de los marcadores. Así el contenido de la cinta en M' quedará:



Para reducir M' a una MT con una sola cinta, M'', se aplicaría la demostración antes dada para la MT de varias pistas: Las distintas pistas 2i se concatenan en una sola pista, introduciendo el marcado de posición de cabeza en cada pista 2i. Así, en nuestro ejemplo partiríamos del contenido en la cinta de M'' con[8]

[8] Aunque aquí se represente una columna con una fila por cada pista, realmente es una sola cinta con todas concatenadas en el orden de las filas

$$\#S_{1,-3}\, S_{1,-2}\, S_{1,-1}@\, S_{1,0}\, S_{1,+1}\, S_{1,+2}\, S_{1,+3}$$
$$\#\, S_{2,-3}\, S_{2,-2}\, @\, S_{2,-1}S_{2,0}\, S_{2,+1}\, S_{2,+2}\, S_{2,+3}$$
$$\#\, S_{3,-3}\, S_{3,-2}\, S_{3,-1}S_{3,0}\, @\, S_{3,+1}\, S_{3,+2}\, S_{3,+3}$$
$$.$$
$$.$$
$$\#\, S_{k-3}\, S_{k-2}\, S_{k-1}S_k\;\; S_{k+1}\, @S_{k+2}\, S_{k+3}\#$$

y después de los sucesivos movimientos de M', que simulan un movimiento de M, M'' tendrá un contenido en su cinta

$$\#S_{1,-3}\, S_{1,-2}\, @\, S_{1,-1}\, S_{1,0}\, S_{1,+1}\, S_{1,+2}\, S_{1,+3}$$
$$\#\, S_{2,-3}\, @\, S_{2,-2}\, Z\;\; S_{2,0}\, S_{2,+1}\, S_{2,+2}\, S_{2,+3}$$
$$\#\, S_{3,-3}\, S_{3,-2}\, S_{3,-1}S_{3,0}\, S_{3,+1}\, @\, S_{3,+2}\, S_{3,+3}$$
$$.$$
$$.$$
$$\#\, S_{k-3}\, S_{k-2}\, S_{k-1}S_k\;\; S_{k+1}\, @S_{k+2}\, S_{k+3}\#$$

Para representar las transiciones de una MT de varias cintas también se utilizarán tablas y diagramas de transiciones:

- Tabla de transiciones.- Tendrá una fila por cada uno de los estados y tendrá una columna por cada tupla del producto cartesiano de los k alfabetos de cinta. Cada entrada contendrá la 2k-tupla con los símbolos a sustituir y el desplazamiento de las cabezas correspondiente.

- Diagrama de transiciones.- El diagrama será el mismo que la MT con una cinta, con la excepción de que la relación entre estados (arista del grafo) se etiqueta con todas las transiciones posibles para esos dos estados en cada una de las cintas, separados por una barra.

3.3.7. MT no determinista (MTN)

Como ya es sabido, el no determinismo de una máquina puede ser de dos tipos: que la máquina no esté totalmente definida y que, para un estado y un símbolo de cinta, la máquina pueda tener más de una opción posible.

Ya se ha dicho que el modelo de la máquina original podría incurrir en el primer tipo de no determinismo y que la consideramos determinista ya que se define con una función (solo una posible transición) y sin estados de rechazo para simplificar su representación. Por tanto, se considerarán no deterministas a las del segundo caso, en las que cada transición, para un estado y un símbolo de cinta, será un conjunto de movimientos posibles, en lugar de un solo movimiento. Así, una máquina no determinista, MTN, para un mismo estado y mismo símbolo de cinta puede no tener transición definida, tener un único movimiento o tener la posibilidad de varios movimientos. En los dos primeros casos se comporta como la MT ya vista, mientras que cuando ocurre el tercero, la máquina **sigue todos los posibles caminos en paralelo** hasta un estado de aceptación. En paralelo significa que no se coge un camino y se sigue hasta que se rechace o se acepte, sino que se producen las configuraciones sucesoras de cada camino a la vez. Si uno de los caminos llega a un estado de rechazo, este camino se desecha y si es el último posible, entonces se rechaza la cadena.

Se dice que una MTN acepta un lenguaje, *si existe algún camino* que lleve desde la configuración inicial a una configuración de aceptación. Observar que se habla de posibilidad de existencia de un camino, no de producción por la función de transición.

La definición formal de la MTN, M, será

$$M = (Q, \Sigma, \Gamma, \sigma, q_0, B, F)$$

donde σ es el conjunto de transiciones

$$\sigma \subset \{Q' \times \Gamma \rightarrow (Q \times \Gamma \times S) \text{ / para todo } Q, \Gamma, S\}$$

Para representar el funcionamiento de M, se representan los posibles movimientos y los posibles caminos mediante el recorrido en amplitud de un grafo: los

nodos serán las configuraciones y las aristas las transiciones. El nodo raíz va a ser la configuración de inicio. Cuando la transición tenga varios posibles movimientos, cada una de las configuraciones resultantes será un nodo hijo del anterior. Los distintos caminos se seguirán en paralelo, desechándose aquel para el que no haya transición posible. Se aceptará la cadena si llegamos por algún camino a un estado de aceptación. Esta aceptación será la de menor profundidad en el grafo.

Para obtener la MT determinista (MTD) M', que simule la MTN, M, se construye una MT de tres cintas (ya sabemos que es equivalente a una MT de una cinta) que va a simular el recorrido en anchura del grafo implícito asociado a las transiciones de M.

Definimos M' = (Q', Σ, Γ', δ', q_0, B, F) donde

- Q' es el conjunto de estados que resulte de la conversión

- Σ, q_0, B y F igual que en M.

- Γ'= Γ ∪ Q ∪ {#,@} .- donde # es un separador de configuraciones y @ es un separador de configuraciones que indica que es la actual.

- δ' ∈ {Q' x Γ → (Q x Γ x S) / para todo Q ,Γ, S }

La primera cinta contendrá la cadena de entrada y no se va a modificar. La segunda cinta contendrá una cola FIFO de configuraciones de la máquina que aun no se han evaluado. La tercera cinta servirá para evaluar la configuración actual de M. El algoritmo de M' realizará la siguiente secuencia:

- Copia en la cinta 1 la cadena de entrada y en las cintas 2 y 3 la configuración inicial de M, esto es, el contenido de la cinta junto con el estado inicial de M. En la cinta 2 se copiará precedida por la izquierda con @ (configuración actual).

- (2) Para la configuración de la cinta 3, se comprueba si es de aceptación, en cuyo caso M' se detiene aceptando la cadena. Si no es de aceptación obtiene de la UC las

transiciones para esta, y guarda las configuraciones resultantes al final del contenido de la cinta 2, precedidas por # como separador de configuraciones.

- En la cinta 2 desmarcamos la configuración actual (sustituimos @ por #) y marcamos como actual la siguiente configuración. Si no hay una configuración al final de la cinta, entonces es que la máquina debe parar rechazando la cadena.

- Borramos el contenido de la cinta 3, copiamos el ella la configuración actual y pasamos al segundo punto anterior (2).

Como ya se ha dicho, el recorrido en anchura simula el funcionamiento de M, en cuyo cas M' aceptará con la misma configuración que aceptaba M, por lo que hemos demostrado que los lenguajes aceptados por una MTN también lo serán por una MT determinista.

Una consecuencia inmediata de la definición de la MTN es que la MT determinista es un caso especial de esta (conjuntos de transiciones de solo un elemento), luego los lenguajes aceptados por la MT determinista, también lo serán por la MTN. Por tanto, MTN y MTD aceptarán exactamente las mismas cadenas.

Para representar una MTN en una tabla de transiciones habrá que modificarla para que en cada entrada de la tabla, en lugar de una tupla con la transición, puede contener varias tuplas. Para hacerlo con el diagrama de transiciones, la diferencia estará en que una arista podrá tener más de una transición en su etiqueta (hay que etiquetarlas de forma distinta a la máquina de varias cintas, por ejemplo en líneas distintas). A la hora de simular el funcionamiento, lo haremos con el grafo antes descrito y realizando el recorrido en anchura.

3.3.8 MT con cintas semi infinitas

Esta máquina es una restricción a la MT original, de manera que ahora la cinta es infinita sólo por el lado derecho, y no por los dos lados. La cadena se introducirá en las casillas más a la izquierda de la cinta y la posición inicial de la cabeza será la primera casilla de la izquierda. Otra restricción de esta máquina es que no está permitido escribir símbolos de casilla en blanco.

Para convertir una MT original M, en una MT de cinta semi infinita M', que descrita formalmente es

$$M' = (Q', \Sigma', \Gamma', \delta', q_0', (B,B), F')$$

lo primero que hay que hacer es partir la cinta de M en dos mitades por la posición indicada por la configuración inicial de M. Después se convierten las dos mitades en una cinta semi infinita de dos pistas, de manera que, si asociamos las posiciones de la cinta de M con un número entero, la pista superior serán las posiciones positivas incluido el cero y el resto en la segunda pista, pero en orden invertido y precedidas por * como marcador de comienzo de cinta. Así, para la cinta de M con los subíndices como posiciones, X como símbolos de cinta y B como símbolo de casilla en blanco

$$\ldots B_{-m}\, B_{-(m-1)} \ldots\, B_{-3}\, B_{-2}\, B_{-1}\, X_0\, X_1\, X_2\, X_3 \ldots\, X_{m-1}\, X_m \ldots B_{m+1}$$

dará como resultado, para M', una cinta de dos pistas semi infinitas de la siguiente forma

X_0	X_1	X_2	X_3	....	X_{m-1}	X_m	B_{m+1} ....
*	B_{-1}	B_{-2}	B_{-3}	....	$B_{-(m-1)}$	B_{-m}	....

El segundo paso es modificar M para convertirla en una máquina que no escriba casilla de espacio en blanco. Esto se hará añadiéndole un símbolo de cinta B' para simular los símbolos de casilla vacía, de manera que una transición de M $(p, X) \rightarrow (q,B,S)$, se conviertirá en $(p, X) \rightarrow (q,B',S)$, lo cual no afectará al funcionamiento de M.

El alfabeto de M' será $\Sigma' = \Sigma \times B$, dado que los símbolos de cinta ahora son duplas: para cualquier elemento $a \in \Sigma$, el equivalente en Σ' es (a,B).

Al principio la cinta en M' está vacía (las dos pistas). Se introduce la entrada a partir de la primera casilla, que serán pares del tipo (a,B), y se posiciona la cabeza sobre la primera columna. Antes de simular los estados de M, se debe marcar el principio de cinta con * en la primera casilla de la segunda pista. Para ello se define Q' como $\{ q_0', q_1' \} \cup \{Q \times P\}$, donde q_0' y q_1' son los estados que marcan el principio de cinta, en la segunda pista, con *, Q es el conjunto de estados de M, y P es una memoria que indica si la máquina se encuentra

en la pista **S**uperior o la **I**nferior; así el contenido de P = {S,I} o, si son las iniciales de las palabras en inglés Up y Down, P = {U,D}.

El alfabeto de cinta de M', serán pares de símbolos de cinta de M al que se le añaden pares de la forma (X,*) para todo X de Γ. Formalmente

$$\Gamma' = (\Gamma \times \Gamma) \cup (\Gamma \times *)$$

Una vez definidos los estados y los símbolos de cinta de M', se puede especificar q_0' y q_1' de manera que, al iniciar la ejecución, las dos primeras transiciones sean

- $((q_0',B),(a, B)) \rightarrow (q1',B), (a, *), D)$ que marca la primera casilla de la cinta inferior con * .

- $((q_1',B),(X, B)) \rightarrow ((q_0,S), (X, B), I)$ para volver a la primera casilla de la cinta superior.

Ahora ya se pueden simular las transiciones de M en estas dos pistas de M', de manera que, para una transición $(p,X) \rightarrow (q,Y,S) \in \delta$:

- Todas las transiciones de M que se realicen a la derecha de la posición inicial de la configuración de inicio serán iguales a las de M', marcando la memoria como S y modificando sólo el símbolo correspondiente a la primera pista.

  $((p,S),(Y,X)) \rightarrow ((q, S) , (Z, X),S)$ para cualquier símbolo de cinta de M.

- Todas las transiciones de M que se realicen a la izquierda de la posición inicial de la configuración de inicio lo serán en dirección contraria, marcando la memoria como I y modificando sólo el símbolo correspondiente a la segunda pista.

  $$((p,I),(X, Y)) \rightarrow ((q, S) , (X, Z), \overline{S})$$

  para cualquier símbolo de cinta de M, siendo $\overline{S}$ el complementario de S (si la transición de M es a la izquierda, en M' es a la derecha y viceversa)

El punto crítico de M' está cuando pasamos de una pista a otra, y esto lo hacemos utilizando el marcador *. Esta situación se da cuando M ha vuelto a la posición inicial, ya sea desde la izquierda (por la segunda pista) o desde la derecha (por la primera pista).

- Para una transición de M de la forma (p,X)→(q,Y,D), en la posición inicial en la que se viene desde izquierda de la cinta, M' estará en un estado (p, I) con símbolo actual (X,*), para cualquier X de símbolos de cinta de M, y la transición cambiará el estado de la memora a S y realiza la sustitución de la transición de M

$$((\mathbf{p, I}), (X,*))\rightarrow((\mathbf{q,S}),(Y,*),D)$$

- Para una transición de M de la forma (p,X)→(q,Y,I), en la posición inicial en la que se viene desde la derecha de la cinta, M' estará en un estado (p, S) con símbolo actual (X,*), para cualquier X de símbolos de cinta de M, y la transición cambiará el estado de la memora a I y realiza la sustitución de la transición de M

$$((\mathbf{p, S}), (X,*))\rightarrow((\mathbf{q,I}),(Y,*),D)$$

Por último el conjunto de estados de aceptación F' de M' serán duplas del producto cartesiano F x P.

Podemos decir que M' simula el funcionamiento de M en todo momento, aceptando la cadena en los mismos estados que lo haría M y no escribiendo en ningún momento el símbolo de casilla en blanco (B, B) que es una de las restricciones que teníamos. Por tanto M' aceptará los mismos lenguajes que acepta M. Al ser una restricción sobre M, cualquier lenguaje aceptado por M' lo será también por M y por tanto aceptarán exactamente los mismos lenguajes.

4.- LENGUAJES DECIDIBLES Y RECONOCIBLES POR UNA MT

El hecho de que la MT que decida (que se pare) si la cadena de entrada pertenece o no al lenguaje depende de este mismo y no de cómo se defina el modelo de la máquina. Si identificamos las características del lenguaje del ejemplo del principio

$$L=\{\omega\$\omega \,/\, \omega \in \{a+b\}^*\}$$

se puede llegar fácilmente a la conclusión de que el conjunto de cadenas aceptadas es un conjunto infinito contable (se puede definir una aplicación sobre el conjunto $\mathbb{N}$ de los número naturales). Así mismo, si se analiza el lenguaje complementario al anterior (las cadenas que no pertenecen al lenguaje), también se podrá demostrar que se trata de un conjunto infinito contable. Por tanto, se puede decir que este lenguaje es un conjunto recursivo o, simplemente, que el lenguaje es recursivo. En el ejemplo propuesto, la MT siempre va a parar: siempre identifica si la cadena pertenece o no al lenguaje. Este comportamiento será siempre el mismo al margen de la variante de MT utilizada.

Pero, recordemos, una MT también puede tener otro comportamiento: entrar en un bucle sin fin. Así, según el comportamiento que tenga una MT se puede afirmar que

- El lenguaje es **decidible** si la MT que lo acepta siempre se para: en estado de aceptación si la cadena es válida y en cualquier otro estado si no lo es (se rechaza la cadena). Los lenguajes decidibles por MT son precisamente los lenguajes recursivos.

- El lenguaje es **reconocible** si la MT que lo acepta se para cuando la cadena pertenece al lenguaje, pero, si la cadena no pertenece al lenguaje, no se sabe si la máquina parará o si entrará en un bucle infinito. Los lenguajes reconocibles por MT son precisamente los lenguajes enumerados recursivamente (RE).

Dentro de la jerarquía de lenguajes, los aceptados por las MT se denominan estructurados por frases. Esto es así porque su estructura gramatical no tiene restricciones (excepto que en el antecedente debe tener algún no terminal). Estos pueden ser decidibles o solo reconocibles por la MT. Como se verá más tarde, existen lenguajes que no son reconocidos por las MT, los cuales no tienen ninguna estructura gramatical. El resto de lenguajes inferiores en la jerarquía (los regulares y los independientes del contexto) son todos decidibles. Para demostrar esto último solo se tendrá que construir una MT que simule el funcionamiento de autómata finito o uno de pila respectivamente.

5. ALGORITMO (y 2)

El hecho de que un problema decidible (un lenguaje decidible) se pueda resolver con una MT que siempre se para, constituye un punto importante en la teoría de la decidibilidad. Es más, **una MT que resuelve un problema decidible es la definición formal de "algoritmo".** De esta forma se puede afirmar que si, para un problema existe un algoritmo que lo resuelve, entonces se dice que el problema (el lenguaje que lo representa) es *decidible* por una MT.

Precisamente aquí es donde empiezan los problemas, ya que hay problemas (lenguajes) para los que no hay un algoritmo que los solucione (que los decida) e incluso hay otros para los que no existe ni siquiera una MT que los reconozca (luego se verá porqué). Una MT que sólo reconozca un lenguaje (un lenguaje RE), soluciona un problema a medias, por lo que realmente no será útil en la resolución de un problema.

Un algoritmo es el procedimiento seguido para resolver un problema por una MT que siempre se para, y ya se ha visto que estos lenguajes son llamados *decididibles.* **Todo lenguaje para el que no haya esta MT, se dice que es indecidible** (ya sea un lenguaje reconocible o no por una MT).

6. CODIFICACIÓN DE LAS MT

Para demostrar que existen lenguajes que no son ni siquiera aceptados por las MT, se va a proceder primero a codificar la entrada para que las cadenas sean binarias, esto es que $\Sigma = \{0, 1\}$ (ya se ha dicho que el formato que tenga la entrada es irrelevante, pues se puede codificar como se precise). Luego se codificará también la descripción de la máquina como una cadena binaria. Esta codificación no es única, pudiendo haber otras variantes. Un detalle que servirá más adelante es que *la longitud del código* resultante de la máquina será *finita,* pues su descripción tiene un número finito de elementos.

Ejemplo: Volvamos a la MT que acepta el lenguaje $L = \{\omega\$\omega \ / \ \omega \in \{a + b\}^* \}$. Lo primero que hacemos es listar los elementos del alfabeto de la máquina y codificarlos en binario: Cada símbolo se codificará con una cadena i ceros, donde i es la posición dentro de la lista ordenada, así, si $\Sigma = \{a,b,\$\}$

Orden	Símbolo	código
1	a	0
2	b	00
3	$	000

Una cadena de entrada se codificará concatenando los códigos de cada símbolo separados por un uno. Esta codificación es válida porque cada símbolo tiene al menos algún cero. Así la cadena abaa$abaa se codificará como 010010101000101001010.

Ahora habría que construir una MT, M', cuya entrada fuese binaria, y que determinase que hay el mismo número de grupos de ceros en el mismo orden y a su vez con el mismo número de ceros (como se ve, el problema ahora es totalmente distinto, aunque equivalente).

--

Una vez obtenida la MT con alfabeto de entrada binario, que además solo tendrá un estado de aceptación (si tuviera más de uno, se realizarían los cambios en la máquina que correspondan) se va a codificar, también en binario, el modelo de MT definido por

$$M= (Q, \Sigma, \Gamma, \delta, q_0, B, q_a)$$

- Código de los estados.- Poner los estados en una lista ordenada, de forma que el *primero sea el estado de inicio y el segundo el de aceptación*. Cada estado se codificará con una cadena i ceros, donde i es la posición dentro de la lista ordenada.

- Símbolos de cinta.- Poner los símbolos de cinta en una lista ordenada, de forma que los tres primeros símbolos de la lista serán 0, 1 y B, si estos existen en la máquina, y a continuación los demás. Cada símbolo de cinta se codificará con una cadena de i ceros, donde i es la posición dentro de la lista ordenada.

- Sentido del movimiento.- Codificar con un cero a la izquierda y con dos ceros a la derecha.

- Transiciones.- Con la codificación de sus elementos, su código resultante se obtendrá concatenando la representación de cada elemento e *intercalando un 1 entre cada uno de ellos*. Esta codificación es válida pues cada elemento está representado al menos con un cero.

- Función de transición.- Se ordenan las transiciones (el orden puede ser arbitrario), y se concatenan, *intercalando dos unos consecutivos entre cada una de ellas*. Esta codificación es válida pues en cada transición nunca va a haber dos unos consecutivos.

  Ejemplo: supongamos la siguiente máquina, que acepta cadenas con solo unos y cadenas con solo ceros:

  M= ($\{q_0, q_1, q_2, q_a\}$, $\{0,1\}$, $\{0,1,B\}$, $\{[(q_0,0) = (q_1,0,D)], [(q_0,1) = (q_2,1,D)], [(q_1,0) = (q_1,0,D)], [(q_2,1) = (q_2,1,D)], [(q_1,B) = (q_a, B, D)]\}$, $[(q_2,B) = (q_a, B, D)]\}$, q_0, B, q_a)

  Primero hacemos una lista ordenada tanto de los estados como de los símbolos de cinta (el orden de los estados no es único):

Estado	Código
q_0	0
q_a	00
q_1	000
q_2	0000
Símbolo	
0	0
1	00
B	000

  Dada esta codificación de elementos, la máquina quedará codificada:

  M= 010100010100 11 010010000100100 11 00010100010100 11 000010010000100100 11 00010001001000100 11 0000100010010001000100

  --

En este ejemplo se puede observar que esta es una de las muchas posibles codificaciones para la misma máquina M. De hecho, el número de codificaciones para una máquina M que acepte alguna cadena será, si esta tiene un estado de aceptación y en la cinta tiene al menos los símbolos 0,1 y B:

(Nº estados-2)! * (Nº símbolos de cinta-3)! * (número de transiciones)!

¿Cómo identificamos cada codificación? Porque plasmar cada cadena binaria es tedioso además de ilegible (ya sea de entrada o la codificación de una máquina). Dado un alfabeto $\Sigma = \{0,1\}$, el conjunto de todas las cadenas binarias posibles (Σ^*) es infinito contable ya que podemos asociar un número natural a cada una de las cadenas, es decir, existe la función ($f: \Sigma^* \rightarrow \mathbb{N}$). Así, se ordenan todas las cadenas de Σ^* primero por longitud y dentro de esta por orden binario, asignando en orden, a cada una, un número natural. Así, dada una cadena ω_i , esta será la i-ésima cadena de Σ^*. Las primeras cadenas corresponderán al número:

λ[9] = 1, 0 =2, 1=3; 00=4; 01=5; 10=6; 11=7; 000=8; 001=9; ….

Para obtener la cadena binaria ω a partir del valor de i, decimos que la cadena i-ésima es el valor decimal de la cadena obtenida de la concatenación de 1 y ω:

$i_{(10} = 1\omega_{(2}$

Dado que cualquier MT se codifica con una cadena binaria, ω_i , entonces se puede asociar a cada codificación un número natural, de manera que podemos hablar de la MT i-ésima, M_i. Así mismo se puede asociar un número natural a cada una de las cadenas de entrada de la máquina. Se puede observar que, al ser cadenas binarias, una cadena de entrada cualquiera con número de orden n, sea la codificación de la máquina M_n.

[9] Símbolo de cadena vacía. Otros autores la denotan con ε

7.- LÍMITE DE LAS MÁQUINAS DE TURING

Si se comparan todas las cadenas de Σ^* con las codificaciones de MT, se puede observar que hay muchas cadenas que no se corresponden con ningún código correcto de la descripción de una MT. En este caso se considera que la cadena es el código de una MT que no hace nada (ya se dijo que era un estado sin transiciones), es decir L(MT) = Ø. De esta forma tenemos tantas MT posibles como números naturales, luego podemos decir que el *número de MT es infinito contable*.

Dado que el conjunto de todos los posibles subconjuntos de cadenas (todos los posibles lenguajes) es el conjunto potencia de Σ^*, del que se sabe que es infinito incontable, se llega a la conclusión de que hay más lenguajes que MT posibles, es decir, habrá lenguajes que una MT no podrá reconocer (aceptar).

En conclusión, vamos a tener problemas que son decidibles (recursivos), problemas que no son decidibles, aunque aceptables por MT (enumerables recursivamente, RE), y problemas indecidibles para los que no existe una MT capaz de aceptarlos (no RE).

Para demostrar la afirmación de que hay más lenguajes que máquinas de Turing, se va a utilizar el método de diagonalización de Cantor. Este matemático lo utilizó para demostrar que el cuerpo de los números Reales era un conjunto infinito incontable. Un sencillo ejemplo: Sean tres cadenas de números decimales de tres dígitos, las cuales se organizan en una lista. Para encontrar una cadena de tres dígitos que no esté en la lista utilizamos el método de la diagonal como se describe a continuación.

- Suponer el conjunto X={0,1,2,3,4,5,6,7,8,9} y, dado un digito y_i, el conjunto Y_i ={y_i}; el conjunto complementario de Y_i será Y_i' = X - Y_i.

- Se coge el primer dígito de la primera cadena, y_1: para la nueva cadena no listada se elige un dígito cualquiera de Y_1'. De esta manera se asegura que la nueva cadena no puede ser igual a la primera.

- Se coge el segundo dígito de la segunda cadena y_2: para la nueva cadena no listada se elige un dígito cualquiera de Y_2'. De esta manera se asegura que la nueva cadena no puede ser igual a la primera.

  …

- Se coge el dígito i de la cadena i-ésima, y_i: para la nueva cadena no listada se elige un dígito cualquiera de Y_i'. De esta manera se asegura que la nueva cadena no puede ser igual a la i-ésima.

De esta forma se ha obtenido una cadena de 3 dígitos que no es igual a ninguna de las existentes en la lista.

Ejemplo: dada la lista 455, 273, 921, encontrar una cadena que no esté en esta lista. Si

- 455 $y_1 = 4$, escogemos y_1'=7
- 273 $y_2 = 7$, escogemos y_2'=2
- 921 $y_3 = 1$, escogemos y_3'=5

La nueva cadena, 725, no está en la lista.

--

Cantor demostró que el intervalo (0,1) es incontable, extrapolando, posteriormente, su resultado a todo el conjunto *R*: Suponiendo cierto que para el intervalo (0,1) tenemos la función f: (0,1) → $\mathbb{N}$, se tendrán n números con n decimales, todos distintos entre sí. Si se aplica el método de diagonalización a la parte fraccionaria, encontraremos al menos un número en el intervalo (0,1) que no estaban en la lista; de esta manera la afirmación de que existe f es falsa, y por lo tanto el conjunto R es infinito incontable.

Utilizando la misma técnica, se va a encontrar un lenguaje, denominado de diagonalización L_d, que no puede ser aceptado por ninguna MT, esto es, que no es RE, de manera que se demostrará que hay más lenguajes que máquinas de Turing.

Para ello listaremos en una columna todas las MT, que hemos asociado con un número natural. El resto de columnas (ω_j) son cada una de las cadenas de Σ^*, asociadas también con un número natural. Así una porción de la lista podría quedar

M_i \ ω_j	…	44	45	46	47	…
…	…	…	…	…	…	…
44	…	**0**	0	1	0	…
45	…	1	**1**	1	0	…
46	…	0	0	**1**	1	…
47	…	0	0	0	**0**	…
…	…	…	…	…	…	…

En este momento se está ya en condiciones de definir lo que es la **secuencia o vector característico** de un lenguaje: será una cadena de unos y ceros, donde la posición j de la cadena resultante, iniciando desde la izquierda, será:

- 1, si la cadena ω_j pertence al lenguaje $L(M_i)$.
- 0, si la cadena ω_j no pertenece al lenguaje $L(M_i)$

Así, cada fila de la tabla será el **vector característico** del lenguaje aceptado por la máquina correspondiente, donde se reflejan las cadenas que pertenecen al lenguaje de cada máquina.

Suponer el lenguaje compuesto por cada una de las cadenas que representan el código de la M_i cuya codificación es aceptada por ellas mismas. Como se puede observar, el vector característico de este lenguaje se corresponde con la diagonal de la tabla (la M_{45} y M_{46} aceptan ω_{45} y ω_{46}, es decir, aceptan su propia codificación, mientras que M_{44} y M_{47} no). Se puede demostrar que la máquina que acepta este lenguaje existe, siendo el vector característico de alguna M_k de la lista.

Para obtener el lenguaje de diagonalización, L_d, se procederá con el método descrito anteriormente (obtiene el complementario del anterior), complementando cada una de las entradas en la diagonal de la lista. De esta forma obtendremos el vector característico de L_d :

- Si ω_j pertenece a $L(M_j)$, entonces ω_j no pertenece a L_d

- Si ω_j no pertenece a $L(M_j)$, entonces ω_j pertenece a L_d

Es decir L_d, es el lenguaje compuesto por cada una de las cadenas que representan el código de la M_i cuya codificación no es aceptada por ella misma.

Con este método se ha encontrado un conjunto de cadenas que no es el vector característico de ninguna MT, es decir, *no hay ninguna MT que acepte L_d*. Para aclarar esta afirmación, suponer que existe una MT, M_k, que acepta el lenguaje L_d. Entonces esta máquina debería estar en la lista. Si la cadena ω_k es el código de M_k, pueden ocurrir dos cosas:

- Que ω_k pertenezca a L_d. En este caso M_k debería aceptar la cadena. Sin embargo, se ha demostrado que L_d no contiene ω_k si existe alguna máquina que acepte su propia codificación, debiendo M_k rechazarla. Luego se produce una contradicción.

- Que ω_k no pertenezca a L_d. En este caso M_k debería rechazar la cadena. Sin embargo, se ha demostrado que L_d contiene ω_k si existe alguna máquina que no acepte su propia codificación, debiendo M_k aceptarla. Luego se produce una contradicción de nuevo.

Como se produce la paradoja de que ω_k deber pertenecer y no pertenecer a la vez a L_d, se concluye que no hay ninguna MT que acepte este lenguaje.

8.- CONJUNTOS RECURSIVOS Y RECURSIVOS ENUMERABLES (y II)

Ya se ha comentado que, para los lenguajes recursivos, la complementación es una operación cerrada. También se ha dicho que si un lenguaje L es recursivo también lo es $\overline{L}$. Para demostrar esto último, basta con construir una MT que decida $\overline{L}$:

Sea L(M) un lenguaje recursivo decidido por M, que será una máquina con una cinta, un estado de aceptación y otro de rechazo. Se puede construir M' que decida $\overline{L}$(M) a partir de M, cambiando la característica de aceptación a rechazo y viceversa. De esta forma, cuando M acepta la cadena, M' siempre parará rechazándola y como M siempre para rechazando la cadena, M' parará aceptándola.

Anteriormente también se afirmó que para que un lenguaje L sea recursivo, tanto L como $\overline{L}$ han de ser recursivamente enumerables. Para demostrar esto, se construirá una MT, M_r, de dos cintas, donde la primera sirve para simular M, que acepta L(M), y la segunda sirva para simular M', que acepta $\overline{L}$(M). M_r simulará en paralelo M y M' (ejecutará alternativamente transiciones de una y otra máquina). Si M acepta la entrada, entonces M_r acepta la entrada. Si M' acepta la entrada, entonces M_r rechaza la entrada. Así M_r siempre parará ante cualquier entrada al problema. Esta demostración también es válida si se parte del hecho de que si $\overline{L}$ es recursivo, tanto L como $\overline{L}$ han de ser recursivamente enumerables.

Hasta ahora se han demostrado los casos en los que el lenguaje y su complementario son simétricos (decidibles por una MT). En este caso se puede considerar que el problema es el mismo.

Sin embargo, también hay casos en los que no lo son, como se demostró en con el lenguaje de diagonalización L_d y su complementario. En este caso cada lenguaje es un problema diferente. Es más, el complementario a L_d es RE (es aceptado por alguna M_k), mientras que L_d no, de lo que se demuestra que la complementación para RE no es cerrada.

En resumen, podemos relacionar un lenguaje y su complementario de la siguiente manera:

- L y $\overline{L}$ son recursivos.

- L y $\overline{L}$ son no RE. Por tanto no se pueden aceptar con una MT.
- L es RE pero $\overline{L}$ es no RE.
- $\overline{L}$ es RE pero L es no RE.

A modo de ilustración, L_d y su complementario están en el tercer caso (recordar que hemos complementado la diagonal de la tabla),

9.- MAQUINA UNIVERSAL DE TURING

Suponer el siguiente problema: determinar si una MT cualquiera acepta una cadena de entrada. Para ello se va a utilizar como entrada a la MT, que resuelve este problema, el par (M, ω): la codificación de la máquina M que acepta la cadena ω. Este par, codificado como lo hemos hecho antes, es la concatenación del código de M y la cadena ω separados por tres unos consecutivos. Esta separación es correcta dado que la codificación de la máquina terminará siempre con algún cero, y además no va a tener más de dos unos consecutivos; así mismo la cadena ω no va a tener nunca dos unos consecutivos.

El lenguaje que acepta esta máquina será el conjunto de todas las codificaciones de máquinas de Turing junto con todas las cadenas que aceptan; así, el problema se reduce a resolver si un par (M, ω) pertenece a este nuevo **lenguaje**, que llamamos **universal**

$$L_u = \{(M, \omega) / M \text{ es una MT que acepta } \omega\}$$

La MT, U, que acepta L_u se la llama **máquina de Turing Universal**, y recibe este nombre debido a que es una máquina que es capaz de simular cualquier otra MT, M, con la cadena entrada, a partir de la descripción codificada de la misma. Este es el motivo por el que se dice que U es una máquina programable, pues a la entrada tiene el código de cualquier MT con sus datos de entrada.

Esta máquina, podrá tener el siguiente comportamiento, mientras simula M:

- Parará aceptando si M para aceptando o
- Parará rechazando si M para rechazando o
- Entrará en un bucle si M entra en un bucle. Este hecho, el de no saber si la máquina parará con una entrada determinada, a veces se le llama **problema de la parada[10].**

Por este comportamiento último, es por el que, intuitivamente, se puede observar que L_u es indecidible. Sin embargo, este lenguaje sí es RE, ya que L_u es precisamente lo que representa toda la lista de máquinas con todas las cadenas que acepta cada una. Luego se puede construir una MT, U, de tres cintas que simule cualquier otra MT, M, con una cadena de entrada determinada:

- Cinta 1.- Se almacena el par (M, ω).
- Cinta 2.- Simula el contenido de la cinta de M (en el mismo código binario utilizado para codificar la máquina, es decir, un cero para el cero, dos ceros para el 1, tres ceros para B,….). El código de cada símbolo se precede con un uno. Esto es correcto porque ningún símbolo de cinta de M se va a codificar con un uno.
- Cinta 3.- Contendrá el estado en el que se encuentra M. Al igual que antes, con el mismo código que la máquina.

Esta máquina tendrá el siguiente comportamiento:

- Primero se comprueba el tipo de máquina que se va a simular. Si el código introducido no se corresponde con el código correcto de alguna máquina, la máquina parará rechazando la cadena (ya se dijo que se considera que un código mal formado representa una MT con un estado y ninguna transición).

[10] Como ya se ha dicho, Turing diseñó la MT calculadora con aceptación por parada. Para esta máquina el problema equivalente al de la parada es si la máquina sería capaz de parar para todas las cadenas a su entrada. Dado que el modelo es distinto, el problema es distinto. Más concretamente, a este problema, para U, se le debería llamar el *problema de la aceptación*, pues se trata de un modelo de aceptación de cadenas. Las conclusiones que sacaremos son las mismas.

- Se codifica la cadena de entrada y se introduce en la cinta 2. Por ejemplo, si la cadena de entrada fuera 01101, en la cinta 2 se almacenaría 1010010010100. Un caso especial será el espacio en blanco: U considerará que la casilla en blanco de la cinta 2 es el código de casilla en blanco de M. De esta manera en la cinta nunca habrá tres ceros seguidos, pero sí una casilla en blanco. Colocamos la cabeza de la segunda cinta en la primera casilla en la empieza el código de ω.

- Almacenamos en la cinta 3 el estado inicial de M.

- Se busca en la cinta 1 la transición correspondiente al estado de la cinta 3 y el símbolo actual de la cinta 2, y se aplica:

    - Se sustituye el estado de la cinta 3 por la codificación del siguiente estado (el que marca la transición).

    - Se sustituye el código del símbolo de cinta actual de M en U por el que marque la transición. Cuando hay una sustitución puede ocurrir que el código del símbolo nuevo tenga una mayor longitud que el código del símbolo a sustituir, en cuyo caso hay que desplazar, en la cinta 2, el resto de la cadena a la derecha las casillas que sean necesarias (igualmente, si el código del símbolo nuevo es de menor longitud, habrá que desplaza a la izquierda el resto de la cadena, y evitar así espacios en blanco). Estos desplazamientos se podrían evitar si se codifican los estados y los símbolos como cadenas binarias de longitud fija (si x es el número de estados/símbolos, $x \geq 2^n$, donde n es el número de bits para un estado/símbolo).

    - Se realiza el movimiento que marque la transición, posicionando la cabeza de la cinta 2 al principio del código del siguiente símbolo (en el 1 que lo marca).

- Si no hay transición definida en M, esta para, luego U se para sin aceptar.

- Si M acepta ω, entonces U terminará en su estado de aceptación.

Una vez que se ha demostrado que hay una MT que simula cualquier MT que acepta una cadena, y por tanto que es RE, ahora hay que demostrar que L_u es indecidible, es decir que no hay una MT para el lenguaje $\overline{L_u}$. De forma intuitiva se puede observar que $\overline{L_d}$ es un subconjunto de L_u. Por esto mismo, L_d estará incluido en $\overline{L_u}$; sabemos que no hay ninguna MT que acepte L_d, por lo tanto no habrá ninguna máquina que acepte $\overline{L_u}$.

Para demostrar esta afirmación, suponer que L_u es recursivo; esto es lo mismo que suponer que existe una MT, M_k, que con una entrada (M, ω) acepta L_u, aceptando si la cadena está en el lenguaje y rechazando si no lo está.

Tomando como base M_k, se va a construir una MT, $\overline{M_k}$, que acepte $\overline{L_u}$ con una entrada (M, ω), igual que hicimos antes: Cuando M_k acepta la cadena de entrada, $\overline{M_k}$ la rechazará y cuando M_k rechace la cadena, $\overline{M_k}$ la aceptará. A continuación se modifica $\overline{M_k}$, para obtener $\overline{M_k}$', que en lugar de tener como entrada el par (M, ω) tenga el par (M,M), es decir, una M que acepte su propia codificación. Así, $\overline{M_k}$' rechazará el par (M,M) si M acepta su propia codificación y aceptará el par (M,M) si M rechaza su propia codificación. Como se puede observar, $\overline{M_k}$' es la máquina que acepta L_d, la cual ya se demostró que no existía. Es fácil ver que si se introduce en $\overline{M_k}$' su propia codificación como entrada, debería rechazar el par ($\overline{M_k}$', $\overline{M_k}$') si $\overline{M_k}$' acepta su propia codificación (lo que es una contradicción) y, por otro lado, debería aceptar el par ($\overline{M_k}$', $\overline{M_k}$') si $\overline{M_k}$' rechaza su propia codificación (lo que también es una contradicción). Al demostrar que $\overline{M_k}$' no existe, se está demostrando que $\overline{M_k}$ tampoco existe y por tanto la afirmación de que existe M_k es falsa. Es decir, el par ($\overline{M_k}$', $\overline{M_k}$') debe pertenecer y no pertenecer a $\overline{L_u}$, con lo que se concluye que **no hay una MT que acepte** $\overline{L_u}$.

Como corolario a toda esta demostración, y al ser $\overline{L_d}$ un subconjunto de L_u, se puede afirmar que, como L_u es RE, cualquier subconjunto suyo también lo es, entre ellos $\overline{L_d}$.

9.1. LÍMITE DE LOS COMPUTADORES ACTUALES

Se ha demostrado que hay infinitos problemas para los que no se va a tener una MT que los resuelva (ni siquiera los reconozca). También se ha formulado la tesis de Church-Turing, que determina el límite de los computadores actuales. Si esto es así ¿Qué tipo de máquina es

un computador actual? Como ya se habrá supuesto, es una máquina programable en el sentido de la Máquina Universal de Turing. Para demostrar esta afirmación, se procederá como siempre: primero simular una MT con un computador y segundo simular un computador con una MT.

Para el primer paso, si tenemos una MT con cinta semi-infinita, la unidad de control se simula por el procesador, cada uno de los símbolos de cinta se codifican como la información que se permiten guardar en la memoria del computador. Los estados, al ser finitos, se pueden guardar en una tabla (en realidad sería el estado del procesador, que incluye el contador de programa) y las transiciones se simulan mediante un programa. La cinta se simularía con la memoria. El inconveniente de que un computador tiene una memoria finita se soluciona suponiendo que siempre es posible añadirle, de forma indefinida, más memoria. De esta forma un computador simularía el funcionamiento de una MT cualquiera.

El segundo paso, pasar de un computador a una MT de varias cintas, es también casi inmediato. El programa y los datos de entrada al programa que se introducen en la memoria antes de ejecutarse en el computador se pueden codificar como un par (M, ω), donde M es la secuencia de instrucciones que hay que computar y ω son los datos de entrada al programa, almacenándose en la primera cinta. En la segunda cinta almacenaremos las posiciones de memoria a las que va a acceder el programa para su lectura/escritura. En la tercera cinta almacenaremos el contador de programa (el estado), que es el que determina la dirección de la computación. De esta forma la MT simularía el funcionamiento de un computador muy básico. La memoria caché, los registros, la UAL son ampliaciones que se han hecho para mejorar la eficiencia.

Sin embargo, un computador admitirá solo un subconjunto de L_u, exactamente el conjunto de aquellas para las que el par (M, ω) sea la codificación de la MT que acepta un lenguaje recursivo, esto es, solo aquellos problemas para los que pueda existir un algoritmo.

9.2. UN PROBLEMA REAL

La verificación formal de un programa ¿es decidible?. Podemos enunciar este problema de otra manera ¿existe alguna MT que decida si su propia definición es válida o no? Por la demostración de la indecidibilidad de L_u, podemos decir que no.

Otro problema que corrobora esta afirmación es que la lógica de predicados, en la que se basa la verificación formal, tampoco es decidible.

10. DETERMINAR LA INDECIBILIDAD. MÉTODO DE REDUCCIÓN

Hasta ahora hemos utilizado un método laborioso para demostrar que un lenguaje es indecidible. Una manera más simple de hacer esta operación es utilizando el *método de reducción*, el cual está implícito en nuestra manera de pensar a la hora de solucionar ciertos problemas: dado un problema P1, este se reduce a solucionar P2. Es decir, si solucionamos P2, tenemos solucionado P1. De esta manera hemos convertido un problema en otro.

Pongamos un ejemplo:

P1.- Hay hambre en el mundo.

P2.- Hay que redistribuir parte de la riqueza

El problema del hambre en el mundo se reduce a redistribuir riqueza. P2 es un problema mucho más general, pues además de solucionar el hambre también solucionaría otros más.

Hay que hacer notar que este método no hace referencia a la manera en la que se soluciona P1 o P2, sino que determina como la solución de P2 conduce a solucionar P1. También se puede ver que P2 es un problema más general que P1. De hecho este método no funcionaría en el sentido inverso: siempre se reduce un problema a otro más general.

Otro ejemplo más mundano: El jefe al fin se ha decidido a dar las vacaciones, pero estamos a 30 de julio y queremos ir con la familia, que resulta que está en Australia. El problema entonces se reduce a encontrar un billete de avión a este destino.

En términos de máquinas de Turing, dado un lenguaje L_1, que define un problema, su decisión se reduce a otro lenguaje L_2 que es decidible. Para ello se debe encontrar un algoritmo que convierta una cadena ω de L_1 en otra cadena ω' de L_2 (L_2 puede tener más cadenas aceptables, pero nos interesan las que se pueden obtener por el algoritmo desde L_1), decidiéndose entonces si ω' pertenece a L_2.

Para demostrar la indecidibilidad de un lenguaje L, se va a utilizar la reducción ante dos situaciones distintas:

- Que P1 sea RE, el problema se reduce a otro P2 que también es RE.

- Que P1 no sea RE, el problema se reduce a otro P2 que tampoco es RE.

Demostrar que un lenguaje L_2 es recursivamente enumerable / noRE, se hará por reducción al absurdo. Para ello partimos de un lenguaje L_1 del que conoceremos de antemano su no decidibilidad (normalmente se utilizarán L_u o L_d, aunque puede ser cualquier otro del que se haya demostrado su indecidibilidad). Tenemos, en las dos situaciones antes descritas:

- Que haya que demostrar que L_2 es RE. Para ello partimos de la certeza de que L_1 es RE (por ejemplo L_u) y lo reducimos a L_2. Se supondrá que este último es decidible, es decir, que se puede encontrar una MT que, siendo la entrada una cadena ω de L_1, esta se puede convertir en una cadena ω' de L_2, y decidir si esta última pertenece a L_2 o no. Si se encuentra al algoritmo de conversión, se habrá encontrado una MT que es capaz de reducir L_1 a L_2, y por tanto, la manera de decidir L_1. Pero esto es una contradicción, pues sabemos de antemano que L_1 es RE, por lo que, forzosamente, L_2 no puede ser decidible. Se llega a la conclusión de que L_2 también es RE.

- Que haya que demostrar que L_2 no es RE. Para ello partimos de la certeza de que L_1 no es RE (por ejemplo L_d) y lo reducimos a L_2. Se supondrá que este último es RE,

es decir, que se puede encontrar una MT que, siendo la entrada una cadena ω de L_1, esta se puede convertir en una cadena ω' de L_2, y aceptarla si esta última pertenece a L_2. Si se encuentra un algoritmo de conversión, se habrá encontrado una MT que es capaz de reducir L_1 a L_2, y por tanto la manera de aceptar L_1. Pero esto es una contradicción, pues sabemos de antemano que L_1 no es RE (ninguna MT puede aceptarlo), por lo que, forzosamente, L_2 no puede ser RE. Se llega a la conclusión de que L_2 tampoco es RE.

APENDICE A.- BIBLIOGRAFÍA Y REFERENCIAS

[Alegre et al, 1997] Alegre Gil, C., Martínez Pastor, A., Pedraza Aguilera, M.C. *Problemas de matemática discreta.* 1ª Edición. Valencia (España): Servicio de publicaciones de la Universidad Politécnica de Valencia, 1997. ISBN. 84-7721-495-6

[Brena 2003]. Brena, R. *Autómatas y lenguajes. Un enfoque de diseño*.[en línea]. Campus Monterrey. Tecnológico de Monterrey. 1ª edición. Monterrey (México), 2003 [consulta 20-10-2009]. Disponible en http://homepages.mty.itesm.mx/rbrena/AyL.html. ISBN 970-94484-0-4.

[Brookshear, 1993]. Brookshear, J., G. *Teoría de la computación. Lenguajes formales, autómatas y complejidad*. 1ª edición. México, Addison Wesley Iberoamericana, 1993. ISBN 9789684443846

[de la Fuente, 2010] de la Fuente López, R.,J. *Matemática discreta: Conjuntos, combinatoria y grafos.* [en línea]. Divulgación web, 2010. [consulta 17-07-2010]. *Disponible en:* http://www.innova.uned.es/webpages/aconute/matematicas/documentos/Matematica%20discreta_conjuntos_combinatoria_grafos_v20100712-r.pdf

[Fernández 1997]. Fernández Vindel, J., L. *Computabilidad y complejidad. Resumen informal de conceptos básicos* [en línea]. Universidad Nacional de Educación a Distancia UNED 1997.[consulta 15-10-2002]. Actualmente no disponible. Obtenido en http://www.ia.uned.es/asignaturas/autómatas-2

[Hernández et al 2009] Hernández Fernández, I., Mateos Contreras, C., Núñez Valdés, J. *Diofanto, Hilbert y Robinson: ¿Alguna relación entre ellos?*. Números, revista didáctica de las matemáticas [on-line]. Abril 2009, volumen 70 [consulta 15-12-2009]. Págs 75-87. Publicación disponible en http://www.sinewton.org/numeros, artículo disponible en http://www.sinewton.org/numeros/numeros/70/Volumen%2070.pdf
ISSN: 1887-1984

[Hopcroft 2002] Hopcroft, J., Motwani, R. , Ullman, J.D.. *Introducción a la teoría de Autómatas, Lenguajes y Computación*. 2ª edición. Madrid (España): Pearson Educación S.A, impresión de 2007. ISBN 9788478290567

[Rogers 1967] Rogers, H. *Theory of recursive Functions and effective Computability*. 1ª edición. MIT press, reimpresión de 1987. ISBN 9780262680523

[Sipser 2006] Sipser M. *Introduction to the Theory of Computation*. Second Edition, international edition. USA: Thomson Course Technology Inc 2006. ISBN 9780619217648

[Turing 1936] Turing, A., M. *ON computable numbers with an application to the entsheidugsproblem*. London Math. Soc, Ser. 2, Vol. 43, No. 2198. año 1936. Disponible http://www.thocp.net/biographies/papers/turing_oncomputablenumbers_1936.pdf

Diagramas de transiciones capturados de:

Rodger, S.,H. *Jflap*[programa]. Universidad de Duke. Durham. Versión 7.0. agosto 2009 [consulta 10-10-2009] Disponible en http://www.cs.duke.edu/csed/jflap/