Notas de clase para el curso
Introducción a la Teoría de la Computación
I Semestre 2020
Profesor: Rodrigo De Castro K.

Capítulo 6

Máquinas de Turing

6.1. El modelo estándar de Máquina de Turing

Una *máquina de Turing (MT)*, $M = (Q, q_0, F, \Sigma, \Gamma, \delta)$, consta de seis componentes:

1. Q es el conjunto (finito) de estados internos de la unidad de control.
2. $q_0 \in Q$ es el estado inicial.
3. F es el conjunto de estados finales o de aceptación, $\varnothing \neq F \subseteq Q$.
4. Σ es el alfabeto de entrada.
5. Γ es el alfabeto de cinta, que incluye a Σ, es decir, $\Sigma \subseteq \Gamma$. Los símbolos de Γ que no hacen parte de Σ (o sea, los símbolos en $\Gamma - \Sigma$) se utilizan como símbolos auxiliares.
6. δ es la función de transición de la máquina:

$$\delta : Q \times (\Gamma \cup \{\square\}) \longrightarrow Q \times (\Gamma \cup \{\square\}) \times \{\leftarrow, \rightarrow, -\}.$$

   δ es una función parcial, es decir, puede no estar definida en algunos elementos del dominio. El símbolo $\square$ no hace parte del alfabeto de cinta Γ sino que es un símbolo "externo" que representa una casilla en blanco. Por simplicidad, $\Gamma \cup \{\square\}$ se denotará como $\Gamma_\square$.

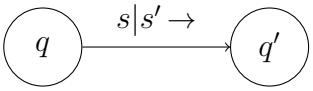
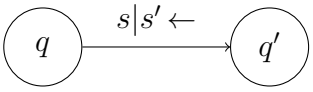
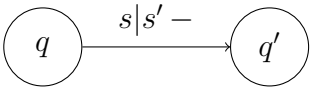
Una máquina de Turing M lee cadenas de entrada $u \in \Sigma^*$ colocadas sobre una cinta infinita en ambas direcciones. Una entrada u se coloca en una porción cualquiera de la cinta y M comienza a leerla escaneando el primer símbolo de u, con la unidad de control

en el estado inicial q_0 Las demás celdas o casillas de la cinta están completamente en blanco.

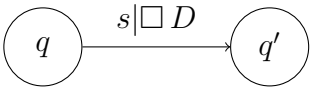
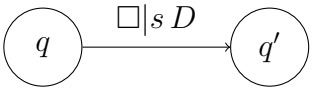
La transición (o instrucción)

$$\delta(q, s) = (q', s', D),$$

donde $s, s' \in \Gamma$, significa: estando en el estado q, escaneando el símbolo s, la unidad de control sobre-escribe s por s', cambia al estado q', y realiza un desplazamiento D, que puede ser $\rightarrow$ o $\leftarrow$ o $-$. Los desplazamientos permitidos y la representación de las instrucciones $\delta(q, s) = (q', s', D)$ en el grafo de estados de la Máquina de Turing se exhiben en la siguiente tabla.

Instrucción	Acción de la unidad de control	Grafo
$\delta(q, s) = (q', s', \rightarrow)$	Se desplaza a la derecha	
$\delta(q, s) = (q', s', \leftarrow)$	Se desplaza a la izquierda	
$\delta(q, s) = (q', s', -)$	Permanece estacionaria	

Hay otros dos tipos de instrucciones permitidas que son: borrar el símbolo actualmente escaneado y escribir un símbolo en una casilla vacía que está siendo escaneada. Se detallan en la siguiente tabla.

Instrucción	Acción de la unidad de control	Grafo
$\delta(q, s) = (q', \square, D)$	Borra el símbolo s y la unidad de control realiza el desplazamiento D, que puede ser $\rightarrow$ o $\leftarrow$ o $-$.	
$\delta(q, \square) = (q', s, D)$	Escribe el símbolo s en la casilla vacía escaneada, y la unidad de control realiza el desplazamiento D, que puede ser $\rightarrow$ o $\leftarrow$ o $-$.	

Como se dijo arriba, $\square$ no hace parte del alfabeto de cinta Γ sino que es un símbolo externo utilizado para presentar las instrucciones $\delta(q, s) = (q', \square, D)$ y $\delta(q, \square) = (q', s, D)$, cuyo significado se acaba de precisar.

En definitiva, toda instrucción básica en una MT es de la forma

$$\delta(q, s) = (q', s', D), \quad \text{donde } s, s' \in \Gamma_{\square};\ q, q' \in Q \text{ y } D \in \{\rightarrow, \leftarrow, -\}.$$

Un detalle importante es que no es necesario definir explícitamente las transiciones λ para máquinas de Turing ya que una instrucción de la forma $\delta(q, s) = (q', s, -)$ permite

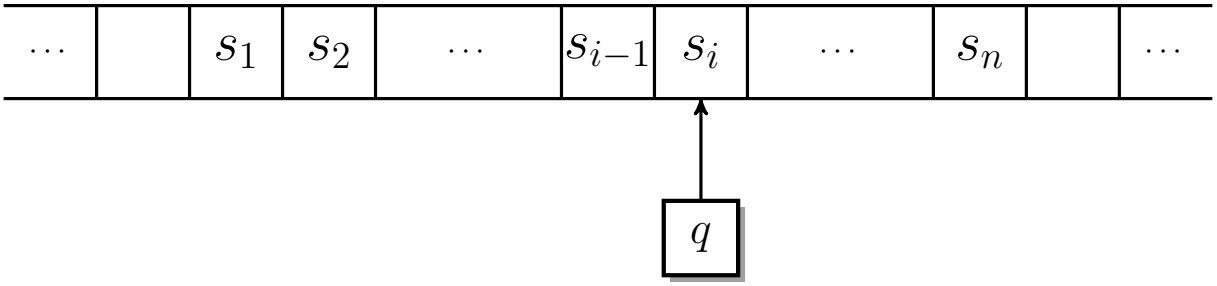
un cambio de estado en la unidad de control sin consumir ningún símbolo ni realizar ningún desplazamiento.

Tal como se ha definido, el modelo estándar de máquina de Turing es determinista y, por tanto, toda cadena de entrada tiene un único procesamiento.

Descripción o configuración instantánea. Es una secuencia de la forma

$$s_1 s_2 \cdots s_{i-1} q s_i \cdots s_n$$

donde los símbolos $s_1, \ldots, s_n$ pertenecen a $\Gamma_\square$ y $q \in Q$. Indica que la unidad de control de M está en el estado q escaneando el símbolo s_i, y todas las casillas a la izquierda s_1 y a la derecha de s_n están enteramente en blanco, tal como se muestra en la siguiente gráfica:



Puesto que toda cadena de entrada es finita, después de que M haya ejecutado un número finito de instrucciones solamente hay una porción finita de la cinta que no está enteramente en blanco; tal porción finita queda representada por la configuración instantánea.

Ejemplos concretos de configuraciones instantáneas son:

$$aabq_2bABa$$
$$q_5ababca$$
$$ab\square\square aabq_0BbA$$
$$\square aX\square Ycq_3Bb\square$$

Una configuración instantánea como $aX\square Ycq_3Bb$ también se puede escribir en la forma $\square aX\square Ycq_3Bb\square$, destacando los símbolos $\square$ en los extremos.

En general, una configuración instantánea es una secuencia de símbolos de la forma uqv donde $u, v \in (\Gamma_\square)^*$ y $q \in Q$. Como caso particular, la configuración instantánea inicial, o simplemente *configuración inicial*, es q_0u, donde $u \in \Sigma^*$ es la cadena de entrada. Otro caso particular importante es el de *configuración de aceptación*, definida más adelante.

Paso computacional. El paso de una configuración instantánea a otra, por medio de una transición definida por δ, se denomina un *paso computacional* y se denota por

$$u_1 q u_2 \vdash v_1 p v_2,$$

donde $u_1, u_2, v_1, v_2 \in (\Gamma_\square)^*$ y $p, q \in Q$. Un ejemplo concreto es

$$abbaq_2ba \vdash abbq_1aca$$

en la cual la máquina ejecutó la instrucción $\delta(q_2, b) = (q_1, c, \leftarrow)$.
La notación

$$u_1qu_2 \stackrel{*}{\vdash} v_1pv_2$$

significa que M puede pasar de la configuración instantánea u_1qu_2 a la configuración instantánea v_1pv_2 en uno o más pasos computacionales. También se utiliza la notación $u_1qu_2 \stackrel{k}{\vdash} v_1pv_2$ para indicar que M pasa de la configuración u_1qu_2 a la configuración v_1pv_2 en exactamente k pasos.

Lenguaje aceptado por una MT. Para simplificar los argumentos sobre máquinas de Turing, en el modelo estándar se supone que la unidad de control siempre se detiene al ingresar a un estado de aceptación. Es decir, no se permiten transiciones $\delta(q, s)$ cuando $q \in F$. La noción de aceptación para máquinas de Turing es más flexible que para autómatas: una cadena de entrada no tiene que ser leída en su totalidad para que sea aceptada; sólo se requiere que la máquina ingrese a un estado de aceptación, y al hacerlo, la unidad de control se detiene inmediatamente. En definitiva, el lenguaje aceptado por una MT $M = (Q, q_0, F, \Sigma, \Gamma, \delta)$ se define como

$$L(M) := \{u \in \Sigma^* : \ q_0u \stackrel{*}{\vdash} vpw, \ p \in F, v, w \in (\Gamma_\square)^*\}.$$

Dicho explícitamente, una cadena de entrada u es aceptada por una MT M si el procesamiento que se inicia en la configuración inicial q_0u termina en una *configuración de aceptación*, es decir, en una configuración de la forma vpw, donde p estado de aceptación. Las cadenas $v, w \in (\Gamma_\square)^*$ que quedan escritas en la cinta cuando M ingresa al estado de aceptación p son irrelevantes.

Si $u \in L(M)$, al leer u, M se detiene en un estado de aceptación. Si $u \notin L(M)$, hay dos situaciones que se pueden presentar al leer u:

1. M se detiene en un estado que no es de aceptación; esto sucede cuando la función de transición $\delta(q, s)$ no está definida para cierta combinación $q \in Q$, $s \in \Gamma_\square$, donde $q \notin F$. Estos son los llamados procesamientos o cómputos abortados.

2. M no se detiene; esto es lo que se denomina un "bucle infinito" o un "ciclo infinito". Esta situación se representa con la notación

   $$q_0u \stackrel{*}{\vdash} \infty$$

   la cual indica que el procesamiento que se inicia en la configuración inicial q_0u no se detiene nunca.

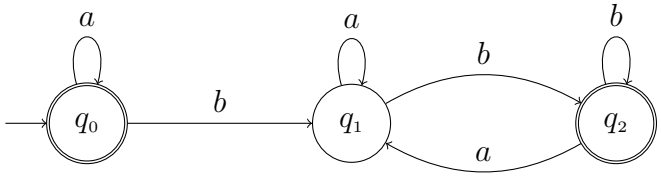
Familias de lenguajes aceptados. Las máquinas de Turing originan las siguientes clases de lenguajes:

1. L es un lenguaje *Turing-aceptable* si existe una MT M tal que $L(M) = L$.
2. L es un lenguaje *Turing-decidible* si existe una MT M tal que $L(M) = L$ y M se detiene con todas las cadenas de entrada.

Obviamente, todo lenguaje Turing-decidible es Turing-aceptable, pero la afirmación recíproca no es (en general) válida. En otras palabras, existen lenguajes que son Turing-aceptables pero no Turing-decidibles como se demostrará en la sección 6.10. Esto permite concluir que el fenómeno de máquinas que nunca se detienen no se puede eliminar de la Teoría de la Computación.

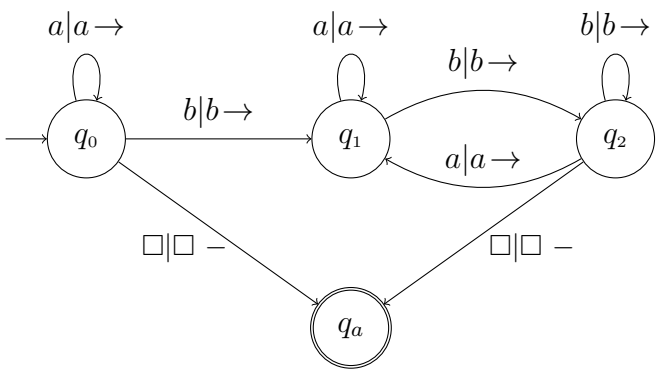
A continuación demostraremos que los lenguajes regulares son Turing-decidibles. Intuitivamente, parece claro que un autómata determinista AFD se puede simular con una MT estándar. No obstante, en un AFD M se permiten transiciones desde estados de aceptación pero en una MT no. La simulación de M por medio de una MT M' consiste en añadir a M un nuevo estado q_a, que será el único estado de aceptación de M', y transiciones con casilla en blanco desde los estados de aceptación originales hasta q_a. Consideremos, por ejemplo, el siguiente AFD M cuyo alfabeto de entrada es $\Sigma = \{a, b\}$.

M:



Las transiciones de la MT M' que simula a M deben declarar explícitamente el desplazamiento a la derecha. Cuando termina de leer una entrada, M detecta una casilla en blanco y se detiene, aceptando si la unidad de control está en los estados q_0 o q_2; la MT M' acepta exactamente las mismas entradas siguiendo las transiciones que llegan a q_a desde q_0 y q_2.

M':



La simulación utilizada en el anterior ejemplo se puede generalizar; esto se hace en el siguiente teorema.

6.1.1 Teorema. Todo autómata finito se puede simular con una MT estándar. En consecuencia, todo lenguaje regular es Turing-decidible.

Demostración. Todo autómata finito (modelo AFD, AFN o AFN-λ) se puede convertir en un AFD $M = (Q, q_0, F, \Sigma, \delta)$ equivalente, usando las técnicas del Capítulo 2. Luego se construye una MT M' que simula a M, añadiendo un nuevo estado q_a, que será el único estado de aceptación de M', y transiciones con casilla en blanco desde los estados de F hasta q_a. En concreto, $M' = (Q', q_0, F', \Sigma, \Gamma, \delta')$ donde

$$\begin{aligned}
&Q' = Q \cup \{q_a\}, \quad q_a \text{ es un estado nuevo,}\\
&\Gamma = \Sigma,\\
&F' = \{q_a\},\\
&\delta'(q, s) = (\delta(q, s), s, \rightarrow), \quad \text{para } q \in Q, s \in \Sigma,\\
&\delta'(q, \Box) = (q_a, \Box, -), \quad \text{para todo } q \in F.
\end{aligned}$$

La MT M' así construida se detiene con cualquier entrada $u \in \Sigma^*$ y $L(M) = L(M')$. Por lo tanto, $L(M)$ es Turing-decidible. □

6.2. Ejemplos de Máquinas de Turing

En esta sección se presentan varios ejemplos concretos de máquinas de Turing. En cada ejemplo se diseña la MT implementando un determinado algoritmo, descrito explícitamente en la forma usual, es decir, como un conjunto de instrucciones que se pueden ejecutar secuencialmente de manera sistemática. En la sección 6.7 se explora a fondo la conexión existente entre la noción intuitiva de algoritmo y los modelos teóricos de computación secuencial, en particular, la máquina de Turing.

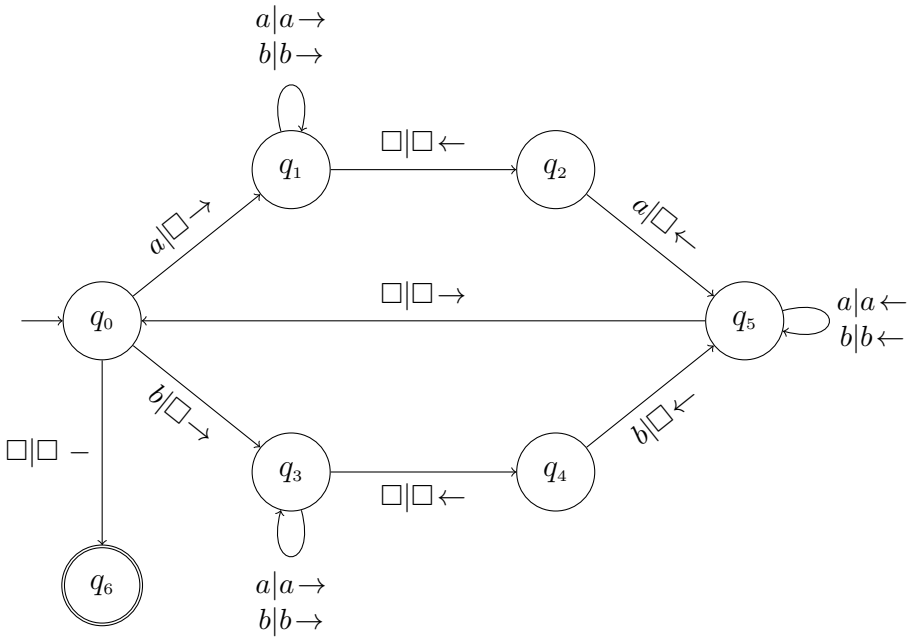
Ejemplo $\Sigma = \{a, b\}$. Diseñar una Máquina de Turing que acepte el lenguaje L de todos los palíndromes de longitud par ≥ 0. $L = \{ww^R : w \in \Sigma^*\}$.

Solución. En el capítulo 4 se construyó un autómata con pila no-determinista para aceptar a L. Es posible aceptar L de manera determinista utilizando el modelo estándar de MT implementado el siguiente sencillo algoritmo.

> Algoritmo: verificar que el primer símbolo de la entrada coincide con el último, el segundo con el penúltimo, y así sucesivamente. Aceptar únicamente cuando la entrada tiene longitud par.

La MT M exhibida a continuación hace la comparación de los símbolos ubicados en extremos simétricos y los va borrando cuando estos coinciden. Para implementar esta idea M no utiliza símbolos auxiliares; en otras palabras, el alfabeto de cinta es simplemente $\Gamma =$

$\{a, b\}$. Es importante resaltar que M está diseñada de tal forma que acepta únicamente palíndromes de longitud par. Cuando una entrada u no es un palíndrome de longitud par, la lectura de u se detiene en el estado q_2 o en el estado q_4. Como M se detiene al leer cualquier entrada, L es un lenguaje Turing-decidible.

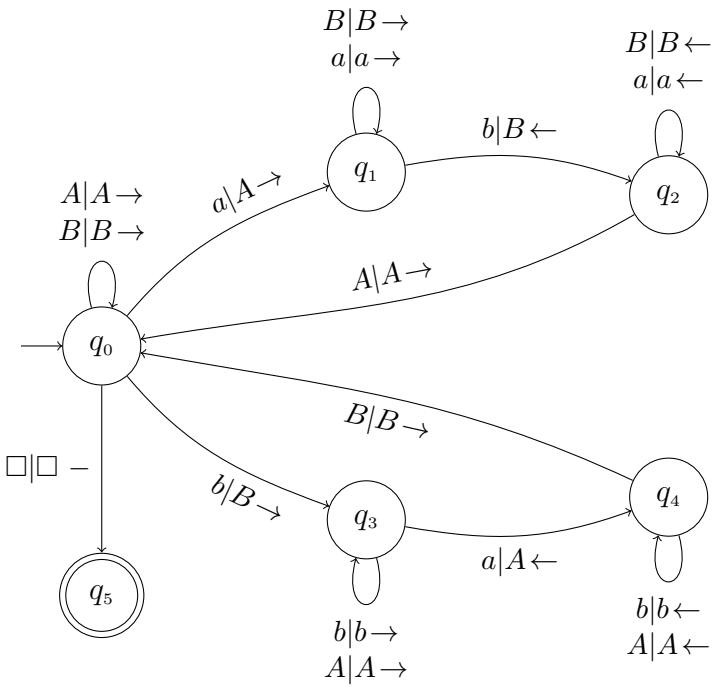


Ejemplo Sea $\Sigma = \{a, b\}$. Diseñar una MT M que acepte el lenguaje de todas las cadenas que tienen igual número de $a$es que de $b$es. Este lenguaje también se ha presentado previamente como $L = \{u \in \Sigma^* : \#_a(u) = \#_b(u)\}$. En el capítulo 4 se diseñaron autómatas con pila para aceptar a L.

Solución. M debe verificar que toda a se corresponde con una única b y viceversa, lo cual se puede conseguir implementando el siguiente algoritmo.

> Algoritmo: si el primer símbolo a la izquierda es a, sobre-escribirla por A y buscar la primera b a la derecha, sobre-escribiéndola por B. Si el primer símbolo a la izquierda es b, sobre-escribirla por B y buscar la primera a a la derecha, sobre-escribiéndola por A. Repetir este ciclo, recorriendo la cadena múltiples veces de izquierda a derecha, hasta que todas las letras minúsculas hayan sido reemplazadas por mayúsculas.

La MT M exhibida a continuación utiliza los símbolos auxiliares A y B, es decir, el alfabeto de cinta es $\Gamma = \{a, b, A, B\}$, y consta de dos ciclos simétricos, dependiendo de si el primer símbolo de la entrada es una a o una b. Al seguir la rama superior de M, la primera a se sobre-escribe por A y M busca la primera b a la derecha, la cual se cambia por B. De manera similar, la rama inferior de M empareja una b (sobre-escribiéndola por B) con la primera a que encuentra a la derecha (la cual cambia por A). Puesto que M se detiene al procesar todas las entradas (algunas aceptadas, otras rechazadas), L es un lenguaje Turing-decidible.



A continuación se presenta el procesamiento completo de la entrada *aababb* utilizando la notación de configuración instantánea.

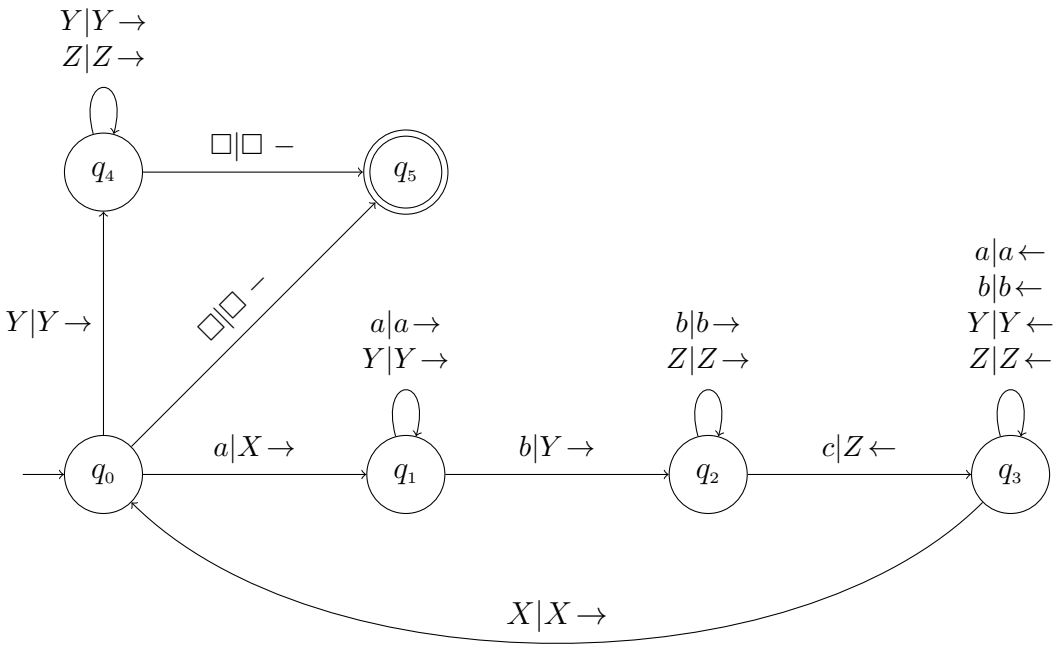
$$q_0aababb \vdash Aq_1ababb \vdash Aaq_1babb \vdash Aq_2aBabb \vdash q_2AaBabb \vdash Aq_0aBabb \vdash AAq_1Babb$$
$$\vdash^2 AABaq_1bb \vdash AABq_2aBb \vdash AAq_2BaBb \vdash Aq_2ABaBb \vdash AAq_0BaBb$$
$$\vdash AABq_0aBb \vdash AABAq_1Bb \vdash AABABq_1b \vdash AABAq_2BB \vdash AABq_2ABB$$
$$\vdash^2 AABAq_0BB \vdash AABABBq_0\square \vdash AABABBq_5\square \text{ (configuración de aceptación).}$$

Ejemplo Sea $\Sigma = \{a, b, c\}$. En este ejemplo se construye una MT M que acepta el lenguaje $L = \{a^n b^n c^n : n \geq 0\}$ y que se detiene al leer todas las entradas. Por consiguiente, L es un lenguaje Turing-decidible aunque no es un LIC, como se demostró en el Capítulo 5; es decir, L no puede ser aceptado por ningún autómata con pila.

La MT M exhibida en la siguiente página implementa el siguiente algoritmo.

> Algoritmo: recorrer la cadena múltiples veces de izquierda a derecha. En cada ciclo completo reemplazar una a por X, una b por Y y una c por Z. Aceptar únicamente cuando todas las letras minúsculas hayan sido reemplazadas por mayúsculas, después de terminar el último ciclo completo.

La MT M utiliza los tres símbolos auxiliares X, Y y Z; es decir, el alfabeto de cinta es $\Gamma = \{a, b, c, X, Y, Z\}$.



La unidad de control cambia la primera a por X y se mueve a la derecha hasta encontrar la primera b, la cual sobre-escribe por una Y. Luego se mueve hacia la derecha hasta encontrar la primera c, la cual se cambia por Z. La unidad de control retrocede entonces hacia la izquierda en busca de la primera X que encuentre en su camino; este retorno se hace en el estado q_3. La máquina avanza luego una casilla hacia la derecha hasta la primera a que quede en la cinta, y todo el ciclo anterior se repite. Si la cadena de entrada tiene la forma requerida, todas las $a$es serán reemplazadas por Xs, las $b$es por Ys y las $c$es por Zs. Al agotarse las $a$es, M detecta la primera Y en el estado q_0, y se mueve hacia la derecha en el estado q_4. Todavía no puede aceptar porque debe verificar que no hay más $b$es, $c$es ni símbolos adicionales en la cadena de entrada. En el estado q_5 M se salta todas las Yes y todas las Zs hasta encontrar la primera casilla en blanco, en cuyo caso ingresa al estado de aceptación q_5.

M está diseñada de tal forma que si una cadena de entrada u no tiene la forma deseada, el procesamiento de u terminará en un estado diferente del estado de aceptación q_5. A continuación procesamos paso a paso la cadena de entrada $w = aabbcc \in L$.

$$q_0aabbcc \vdash Xq_1abbcc \vdash Xaq_1bbcc \vdash XaYq_2bcc \vdash XaYbq_2cc \vdash XaYbq_3bZc$$

$$\vdash^* q_3XaYbZc \vdash Xq_0aYbZc \vdash XXq_1YbZc \vdash XXYq_1bZc$$

$$\vdash XXYYq_2Zc \vdash XXYZq_2c \vdash XXYq_3ZZ \vdash^* Xq_3XYZZ$$

$$\vdash XXq_0YYZZ \vdash XXYq_4YZZ \vdash^* XXYYZZq_4\square$$

$$\vdash XXYYZZ\square q_5\square \text{ (configuración de aceptación).}$$

La cadena de entrada $w = aaabbcc$, que no está en L, se procesa así:

$$\begin{aligned} q_0aaabbcc &\vdash Xq_1aabbcc \vdash Xaaq_1bbcc \vdash XaaYq_2bcc \vdash XaaYbq_2cc \\ &\vdash XaaYbq_3bZc \vdash^* q_3XaaYbZc \vdash Xq_0aaYbZc \vdash XXaq_1YbZc \\ &\vdash XXaYq_1bZc \vdash XXaYYq_2Zc \vdash XXaYZq_2c \vdash XXaYq_3ZZ \\ &\vdash^* Xq_3XaYZZ \vdash XXq_0aYYZZ \vdash XXXq_1YYZZ \\ &\vdash^* XXXYYq_1ZZ \text{ (procesamiento abortado).} \end{aligned}$$

Por otro lado, la cadena $abbcc$, que tampoco está en L, se procesaría así:

$$\begin{aligned} q_0abbcc &\vdash Xq_1bbcc \vdash XYq_2bcc \vdash XYbq_2cc \vdash XYq_3bZc \vdash^* q_3XYbZc \\ &\vdash Xq_0YbZc \vdash XYq_4bZc \text{ (procesamiento abortado).} \end{aligned}$$

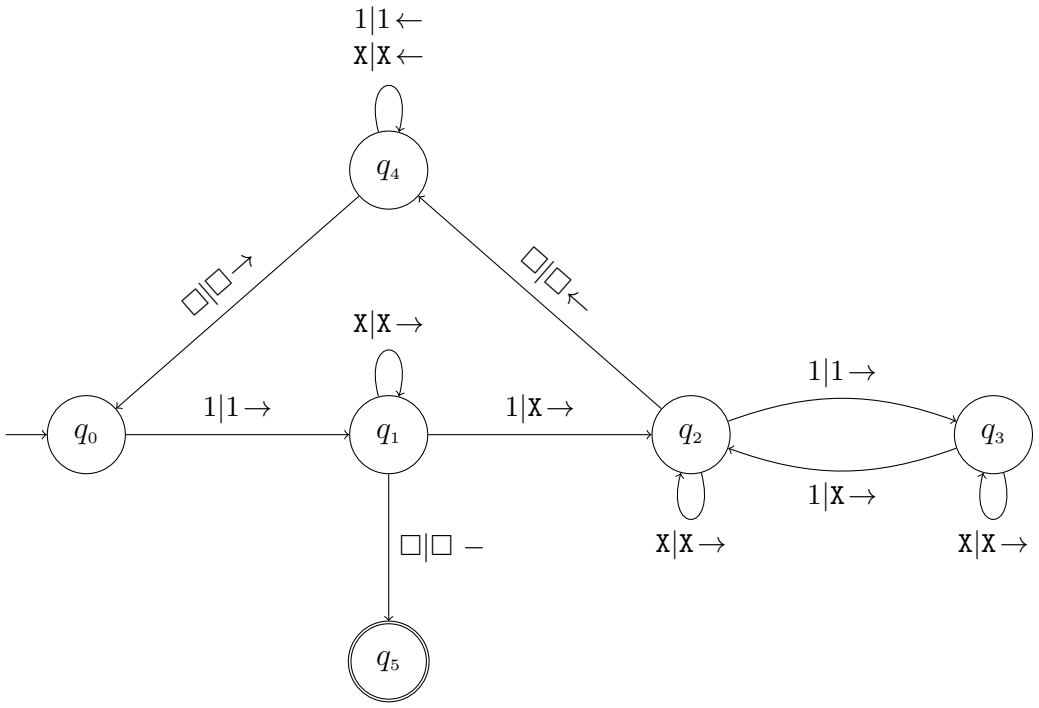
Ejemplo Sea $\Sigma = \{1\}$. Diseñar una máquina de Turing que acepte el lenguaje

$$\{1^{2^n} : n \geq 0\} = \{1, 1^2, 1^4, 1^8, 1^{16}, 1^{32}, \ldots\}.$$

Solución. La idea que hay detrás del algoritmo que se utilizará para construir la MT M es que si un número natural de la forma 2^n se divide sucesivamente por 2 (n veces) se obtiene 1. Esto implica que si en una cadena de entrada de la forma 1^{2^n} se va reduciendo el número de unos a la mitad en n pasos consecutivos se obtiene finalmente la cadena 1. En el siguiente algoritmo los unos que se eliminan en cada iteración son reemplazados por una `X` cada uno.

> Algoritmo: recorrer sucesivamente la entrada de izquierda a derecha reemplazando cada 1 por `X` alternadamente, es decir, un 1 se reemplaza por `X` pero el siguiente no. De esta manera, al completar cada ciclo exactamente la mitad de los unos existentes se han reemplazado por una `X` cada uno. Aceptar cuando quede un único 1 en la cinta.

En la página siguiente se exhibe una MT M que implementa este algoritmo. En un ciclo que comienza en el estado q_0 y retorna a q_0, exactamente la mitad la mitad de los unos son reemplazados por `X`. Tal sustitución se hace en el bucle q_2-q_3. Si la entrada tiene longitud impar o si tiene longitud par que no es de la forma 2^n, el procesamiento se aborta en algún momento en el estado q_3 y la entrada no será aceptada. M se detiene al leer todas las entradas y, en consecuencia, L es un lenguaje Turing-decidible.



A continuación procesamos paso a la cadena 1^{16} destacando las configuraciones instantáneas que completan cada iteración que comienza en q_0 y retorna a q_0. El procesamiento termina en la configuración de aceptación $\texttt{1XXXXXXXXXXXXXXX}q_5\square$

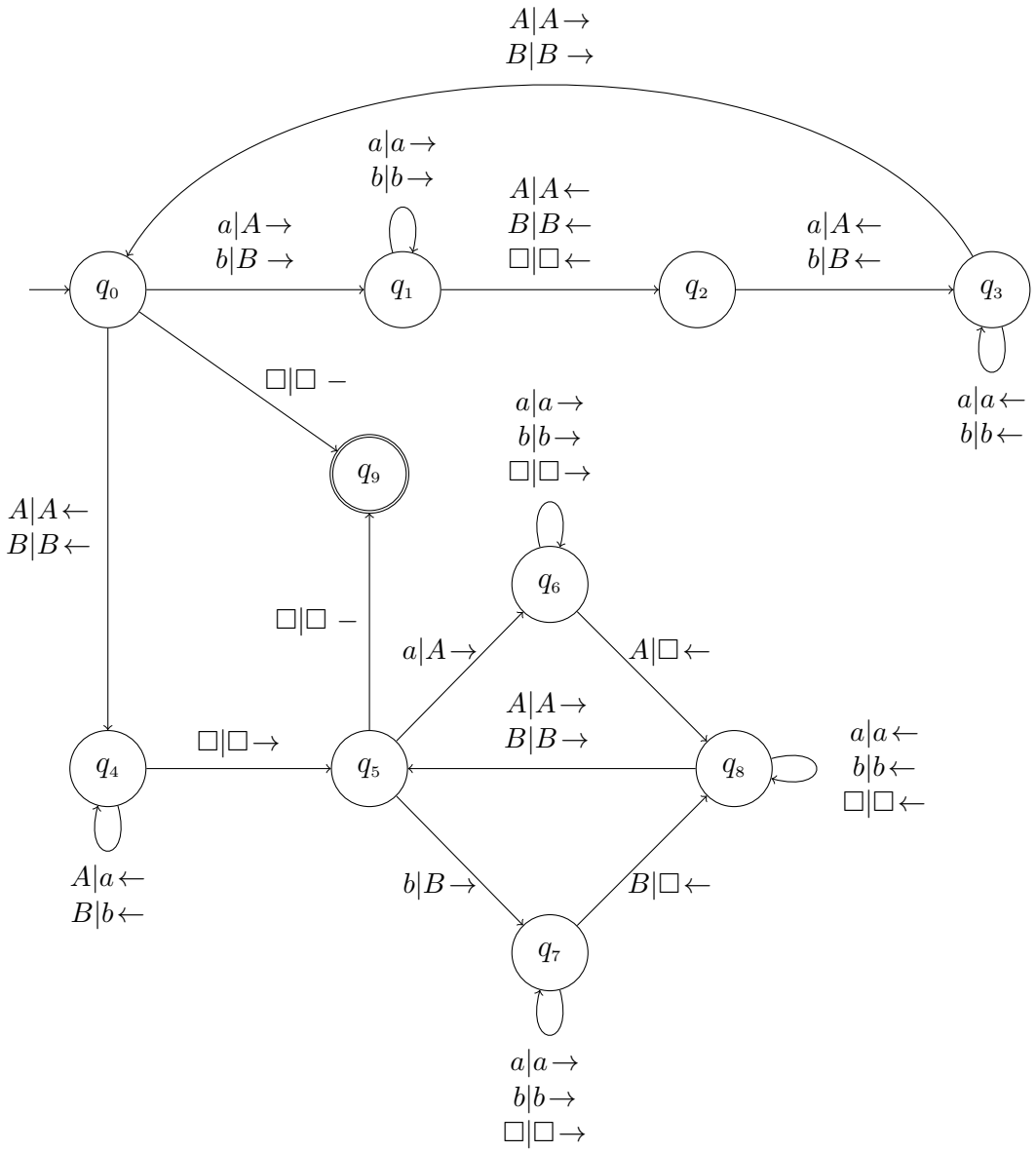
$$
\begin{aligned}
& q_0\texttt{1111111111111111} \\
\vdash^* \; & q_0\texttt{1X1X1X1X1X1X1X1X} \\
\vdash^* \; & q_0\texttt{1XXX1XXX1XXX1XXX} \\
\vdash^* \; & q_0\texttt{1XXXXXXX1XXXXXXX} \\
\vdash^* \; & q_0\texttt{1XXXXXXXXXXXXXXX} \\
\vdash^* \; & \quad \texttt{1XXXXXXXXXXXXXXX}q_5\square
\end{aligned}
$$

Ejemplo Sea $\Sigma = \{a, b\}$. Diseñar una máquina de Turing que acepte el lenguaje $L = \{ww : w \in \Sigma^*\}$.

Solución. La máquina de Turing M exhibida en la página siguiente implementa el algoritmo descrito a continuación.

> Algoritmo: chequear la paridad de las entradas; una cadena de longitud impar debe ser rechazada. Escribir toda entrada u de longitud par como la concatenación de dos subcadenas de la misma longitud; es decir, escribir u en la forma $u = vw$ donde $|v| = |w|$ (esto equivale a ubicar la mitad de la cadena de entrada u). Comparar las dos mitades v y w símbolo a símbolo, leyéndolas de izquierda a derecha, y aceptar únicamente si $v = w$.

Para explicar el funcionamiento de la MT M adoptaremos la siguiente notación: si u es una cadena en Σ^*, entonces U denota la cadena obtenida a partir de u cambiando minúsculas por mayúsculas, o sea a por A y b por B. El alfabeto de cinta utilizado por M es $\Gamma = \{a, b, A, B\}$.



M procesa una entrada $u \in \Sigma^*$ en tres etapas:

1. El ciclo q_0, q_1, q_2, q_3, q_0 sobre-escribe a por A y b por B, ubicando la mitad de la entrada u, cuando esta tiene longitud par: $q_0 u \mid\!\!\!\overset{*}{-} V q_0 W$. Este ciclo sirve además para chequear la paridad de la cadena de entrada u; si u tiene longitud impar, el ciclo se aborta en el estado q_2.

2. En el estado q_4 se sobre-escribe de nuevo la primera mitad V por v y luego M ingresa al estado q_5.

3. El ciclo q_5, q_6, q_7, q_8, q_5 compara las dos mitades v y W, símbolo a símbolo, convirtiendo los símbolos de v en mayúsculas y borrando los correspondientes símbolos de W, si estos coinciden. M acepta únicamente si las dos mitades V y W coinciden, y esto sucede si, estando en el estado q_5, M detecta una casilla en blanco, en cuyo caso ingresa al estado de aceptación q_9.

M se detiene al leer todas las entradas así que L es un lenguaje Turing-decidible. Una cadena de entrada que sea de la forma ww se procesa de la siguiente forma:

$$q_0ww \vdash^* Wq_0W \vdash^* q_4\square wW \vdash^* q_5wW \vdash^* Wq_5\square\square\square\cdots \vdash Wq_9\square.$$

Por ejemplo, la cadena de entrada $abaababaab$, que es de la forma ww, con $w = abaab$, se procesaría de la siguiente forma:

$$\begin{aligned} q_0abaababaab &\vdash^* ABAABq_0ABAAB \vdash ABAAq_4BABAAB \vdash^* q_4\square abaabABAAB \\ &\vdash q_5abaabABAAB \vdash^* Aq_5baab\square BAAB \vdash^* ABq_5aab\square\square AAB \\ &\vdash^* ABAq_5ab\square\square\square AB \vdash^* ABAAq_5b\square\square\square\square B \vdash^* ABAABq_5\square\square\square\square\square \\ &\vdash ABAABq_9\square\square\square\square\square \text{ (configuración de aceptación).} \end{aligned}$$

Ejercicios de la sección 6.2

Sea $\Sigma = \{a, b, c\}$. Diseñar Máquinas de Turing, modelo estándar, que acepten los siguientes lenguajes. Presentar cada MT por medio de un grafo de estados.

① $L = \{a^n b^{n+1} c^{n+2} : n \geq 0\}$.

② $L = \{a^n b^{2n} c^n : n \geq 0\}$.

③ $L = \{a^k b^m c^n : 0 \leq k \leq m \leq n\}$.

④ $L = \{a^k b^m c^n : 0 \leq k < m < n\}$.

⑤ $L = \{a^k b^m c^n : k \geq m \geq n \geq 0\}$.

⑥ $L = \{a^m b^n c^n : m, n \geq 1,\ m > n\}$.

⑦ $L = \{a^m b^n c^n : m, n \geq 1,\ n > m\}$.

⑧ $L = \{a^m b^n c^n : m, n \geq 1,\ m \neq n\}$.

⑨ $L = \{a^m b^n a^m c^n : m, n \geq 1\}$.

6.3. Variaciones del modelo estándar de MT

El modelo estándar de máquina de Turing es bastante robusto en el sentido de que admite numerosas variaciones que no alteran las familias de lenguajes aceptados. En esta sección y en las siguientes consideraremos algunos modelos equivalentes al modelo estándar.

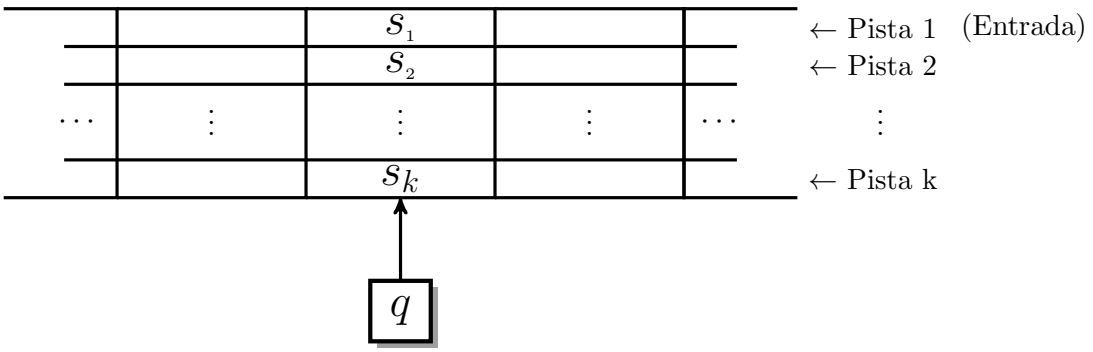
6.3.1. Estado de aceptación único

Una máquina de Turing se puede convertir, sin alterar el lenguaje aceptado, en una máquina con un único estado de aceptación. Esto es posible porque en la definición del modelo estándar se ha exigido que una MT siempre se detenga cuando la unidad de control ingresa a un estado de aceptación. Si una MT tiene varios estados de aceptación, podemos modificarla cambiando cada transición de la forma $\delta(q, s) = (p, s', D)$, donde $s, s' \in \Gamma_\square$ y p es un estado de aceptación original, por la transición $\delta(q, s) = (q_a, s', D)$, en la que q_a es el único estado de aceptación de la nueva máquina. Los estados de aceptación originales se eliminan completamente.

De esta manera, se puede considerar que toda máquina de Turing es de la forma $M = (Q, q_0, q_a, \Sigma, \Gamma, \delta)$, con un único estado inicial y un único estado de aceptación.

6.3.2. Máquina de Turing con cinta dividida en pistas

En el modelo multi-pista, la cinta está dividida en un número finito k de pistas, como se muestra en la siguiente figura:



Inicialmente, la cadena de entrada se coloca en la primera pista, llamada pista de entrada, y las demás celdas, en todas las pistas, están en blanco. En un paso computacional, la unidad de control cambia simultáneamente el contenido de las k pistas de la casilla escaneada y realiza luego uno de los desplazamientos $\rightarrow$, $\leftarrow$ ó $-$. La función de transición adquiere la siguiente forma:

$$\delta(q, (s_1, s_2, s_3, \ldots, s_k)) = (q', (s'_1, s'_2, s'_3, \ldots, s'_k), D)$$

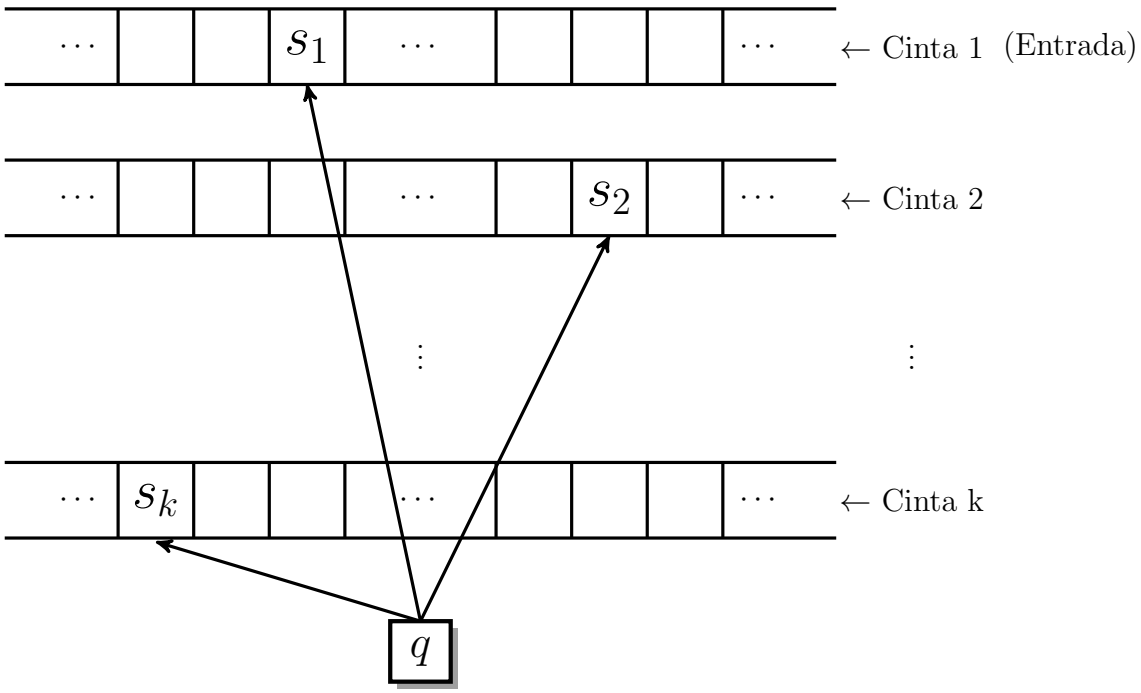
donde los s_i y los s'_i son símbolos pertenecientes a $\Gamma_\square$ y D es uno de los posibles desplazamientos $\leftarrow$, $\rightarrow$ ó $-$.

Simulación. Las máquinas de Turing que actúan sobre una cinta dividida en k pistas aceptan los mismos lenguajes que las MT estándares. Para concluir tal afirmación, basta considerar el modelo multi-pista como una MT normal en la que el alfabeto de cinta está formado por el conjunto de k-uplas $(s_1, s_2, \ldots, s_k)$, donde los $s_i \in \Gamma_\square$. Es decir, el nuevo alfabeto de cinta es el producto cartesiano $(\Gamma_\square)^k = \Gamma_\square \times \Gamma_\square \times \cdots \times \Gamma_\square$ (k veces).

Ejemplo Las pistas se usan por lo general para señalar con "marcas" o "marcadores" ciertas posiciones en la cinta. La MT diseñada en el ejemplo de la sección 6.1 para aceptar el lenguaje $L = \{a^n b^n c^n : n \geq 0\}$ utiliza implícitamente la idea de marcadores: reemplazar las as por Xs, las $b$es por Ys y las $c$es por Zs no es otra cosa que colocar las marcas X, Y y Z en las posiciones deseadas. Estas marcas se pueden colocar en una pista diferente, sin necesidad de sobre-escribir los símbolos de la cadena de entrada.

6.3.3. Máquina de Turing con múltiples cintas

En el modelo multi-cinta hay k cintas diferentes (k finito, $k \geq 2$), cada una dividida en celdas o casillas, como se muestra en la siguiente figura.



La unidad de control tiene k "visores" que escanean una casilla en cada cinta. Inicialmente, la cadena de entrada se coloca en la primera cinta, llamada cinta de entrada, y las demás cintas están enteramente en blanco. En un paso computacional, la unidad de control cambia el contenido de la casilla escaneada en cada cinta y realiza luego uno de los desplazamientos $\rightarrow$, $\leftarrow$ o $-$. Esto se hace de manera independiente en cada cinta; los k visores actúan independientemente en cada cinta. La función de transición tiene la

siguiente forma:

$$\delta(q,(s_1,s_2,s_3,\ldots,s_k)) = \big(q',(s'_1,D_1),(s'_2,D_2),(s'_3,D_3),\ldots,(s'_k,D_k)\big),$$

donde los s_i y los s'_i son símbolos pertenecientes a $\Gamma_\square$, y cada D_i es un desplazamiento $\rightarrow$, $\leftarrow$ ó $-$.

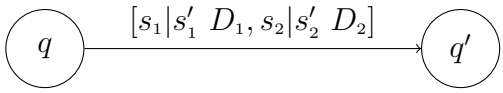
Simulación. A pesar de que el modelo multi-cinta parece, a primera vista, más poderoso que el modelo estándar, resulta que una MT con múltiples cintas se puede simular con una MT estándar. En otras palabras, si una MT multi-cinta acepta un lenguaje L, es posible construir una MT estándar que acepta el mismo lenguaje L. A continuación bosquejaremos el procedimiento de simulación.

Una MT con k cintas se simula con una MT que actúa sobre una única cinta dividida en $2k+1$ pistas. Cada cinta de la máquina multi-cinta da lugar a dos pistas en la máquina simuladora: la primera simula la cinta propiamente dicha y la segunda tiene todas sus celdas en blanco, excepto una, marcada con un símbolo especial X, que indica la posición actual del visor de la máquina original en dicha cinta. La pista adicional de la máquina simuladora se utiliza para marcar, con los símbolos especiales Y y Y', las posiciones más a la izquierda y más a la derecha de la unidad de control en la máquina original. Para simular un solo paso computacional, la nueva máquina requiere hacer múltiples recorridos a izquierda y a derecha, actualizando el contenido de las pistas y la posición de los marcadores X, Y y Y'.

Grafos. Es posible utilizar grafos de estados para MT multi-cintas. Por ejemplo, para una máquina con dos cintas, una instrucción de la forma

$$\delta(q,(s_1,s_2)) = \big(q',(s'_1,D_1),(s'_2,D_2)\big),$$

se presenta en el grafo como

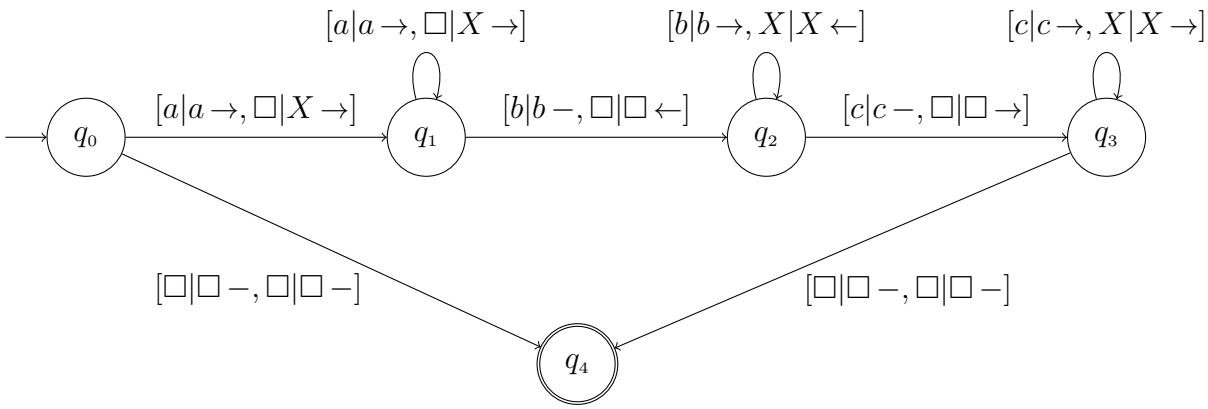


Cada instrucción se encierra entre paréntesis angulares, y las acciones en cada cinta se separan con comas, con la primera cinta a la izquierda y la segunda a la derecha.

Ejemplo Diseñar una MT con dos cintas que acepte el lenguaje $L = \{a^n b^n c^n : n \geq 0\}$.

Solución. Para aceptar este lenguaje se puede diseñar una MT con dos cintas más sencilla y eficiente que la MT estándar presentada en el tercer ejemplo de la sección 6.2 (página 164). La idea es copiar en la segunda cinta una X por cada a leída, y verificar luego hay tantas Xs como $b$es y $c$es. Cuando lee las $b$es, la unidad de control avanza hacia la derecha en la primera cinta y hacia la izquierda en la segunda. Cuando lee las $c$es se avanza hacia la derecha en ambas cintas. Si la cadena de entrada tiene la forma $a^n b^n c^n$, se detectará simultáneamente el símbolo en blanco $\square$ en ambas cintas, y la cadena se aceptará.

Se utilizan cinco estados para implementar esta idea; el grafo de M es:



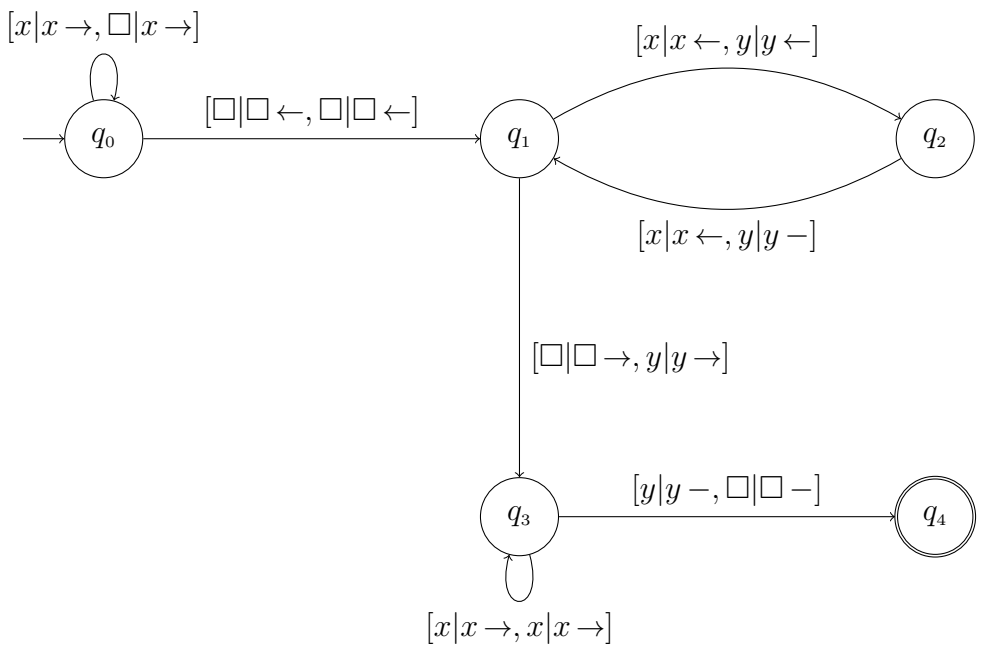
M también se puede presentar por medio de su función de transición:

$$\begin{aligned}
&\delta(q_0,(a,\square)) = (q_1,(a,\rightarrow),(X,\rightarrow)),\\
&\delta(q_1,(a,\square)) = (q_1,(a,\rightarrow),(X,\rightarrow)),\\
&\delta(q_1,(b,\square)) = (q_2,(b,-),(\square,\leftarrow)),\\
&\delta(q_2,(b,X)) = (q_2,(b,\rightarrow),(X,\leftarrow)),\\
&\delta(q_2,(c,\square)) = (q_3,(c,-),(\square,\rightarrow)),\\
&\delta(q_3,(c,X)) = (q_3,(c,\rightarrow),(X,\rightarrow)),\\
&\delta(q_3,(\square,\square)) = (q_4,(\square,-),(\square,-)),\\
&\delta(q_0,(\square,\square)) = (q_4,(\square,-),(\square,-)).
\end{aligned}$$

Ejemplo Diseñar una MT con dos cintas que acepta el lenguaje $L = \{ww : w \in \Sigma^*\}$ sobre el alfabeto $\Sigma = \{a, b\}$.

En la sección 6.2 se presentó una MT estándar para aceptar este lenguaje. Con dos cintas se puede diseñar una máquina más sencilla y eficiente. La idea es copiar la entrada en la segunda cinta y luego retroceder a la izquierda avanzando dos casillas en la primera cinta pero solo una casilla en la segunda. Cuando la unidad de control lee una casilla en blanco en el extremo izquierdo de la primera cinta, se detecta la mitad de la entrada en la segunda cinta. Finalmente, se avanza a la derecha simultáneamente en ambas cintas verificando que los símbolos coinciden uno por uno.

Para presentar la MT se utilizará la siguiente convención: $x, y \in \{a, b\}$. Las letras x, y no son símbolos auxiliares de máquina de Turing sino son variables externas que representan los símbolos de entrada a y b. Con esta convención, el bucle sobre el estado q_0 representa dos instrucciones, una cuando x es a y otra cuando x es b. La transición $[x|x\leftarrow, y|y\leftarrow]$ entre los estados q_1 y q_2 representa cuatro instrucciones teniendo en cuenta las posibles combinaciones de valores para x y y; lo mismo sucede con la instrucción $[x|x\leftarrow, y|y-]$ entre q_2 y q_1. Cuando entrada tiene longitud impar, la máquina se detiene en algún momento estando en el estado q_2. El alfabeto de cinta de esta máquina es simplemente $\{a, b\}$.



A continuación ilustramos el procesamiento de la cadena de entrada *baabbaab* (aceptada) utilizando la notación de configuración instantánea para máquinas de Turing con dos cintas, y resaltando el funcionamiento del bucle q_1-q_2.

$$
\begin{aligned}
(q_0,\ \underline{b}aabbaab,\ \underline{\square})\ \vdash^*\ (q_1,\ baabbaa\underline{b},\ baabbaa\underline{b})\ &\vdash^2\ (q_1,\ baabb\underline{a}ab,\ baabba\underline{a}b)\\
&\vdash^2\ (q_1,\ baa\underline{b}baab,\ baabb\underline{a}ab)\\
&\vdash^2\ (q_1,\ b\underline{a}abbaab,\ baab\underline{b}aab)\\
&\vdash^2\ (q_1,\ \underline{\square}baabbaab,\ baa\underline{b}baab)\\
&\vdash\ (q_3,\ \underline{b}aabbaab,\ baab\underline{b}aab)\\
&\vdash^*\ (q_4,\ baab\underline{b}aab,\ baabbaab\underline{\square}).
\end{aligned}
$$

Ejercicios de la sección 6.3

① Sea $\Sigma = \{a, b, c\}$. Diseñar Máquinas de Turing con dos o más cintas que acepten los siguientes lenguajes. En cada caso, explicar brevemente el plan utilizado en el diseño y presentar la MT ya sea por medio de una lista de transiciones o por medio de un grafo de estados.

(i) $L = \{a^n b^{2n} c^n : n \geq 0\}$.

(ii) $L = \{a^k b^m c^n : 0 \leq k \leq m \leq n\}$.

(iii) $L = \{a^k b^m c^n : k \geq m \geq n \geq 0\}$.

(iv) $L = \{a^m b^n c^n : m, n \geq 1,\ m > n\}$.

(v) $L = \{a^m b^n c^n : m, n \geq 1,\ n > m\}$.

(vi) $L = \{a^m b^n a^m c^n : m \neq n\}$.

(vii) $L = \{a^m b^n a^m c^n : m, n \geq 1\}$.

(viii) $L = \{u \in \Sigma^* : \#_a(u) = \#_b(u) = \#_c(u)\}$.

② Sea $\Sigma = \{a, b, c, d\}$. Diseñar una Máquina de Turing con dos o más cintas que acepte el lenguaje $L = \{a^n b^n c^n d^n : n \geq 0\}$.

6.4. Funciones Turing-computables

Como las máquinas de Turing tienen la capacidad de transformar entradas en salidas se pueden utilizar como mecanismos para calcular funciones. Esta noción se precisa en las siguientes definiciones.

6.4.1 Definicion. Sean Σ y Γ dos alfabetos. Una función $f : \Sigma^* \longrightarrow \Gamma^*$ cuyo dominio, denotado $\mathrm{dom}(f)$, es un subconjunto propio de Σ^* se dice que está parcialmente definida, o que es una función parcial. Si $\mathrm{dom}(f) = \Sigma^*$ se dice que f está totalmente definida o que es una función total.

Una función parcial o total $f : \Sigma^* \longrightarrow \Gamma^*$, con dominio $\mathrm{dom}(f)$, es *Turing-computable* o *Turing-calculable* si existe una MT estándar $M = (Q, q_0, q_a, \Sigma, \Gamma', \delta)$ tal que, para toda cadena $u \in \Sigma^*$,

$$u \in \mathrm{dom}(f) \implies q_0 u \vdash^* q_a v, \quad \text{donde } v = f(u).$$

Es decir, con entrada u, M produce la salida $f(u)$. El estado q_a es un estado de aceptación en el sentido usual, pero aquí representa el estado en el cual se detiene M para producir salidas. Como M es una MT estándar, no se permiten transiciones desde el estado q_a. Nótese que M debe detenerse en la configuración $q_a v$, o sea, la unidad de control debe estar escaneando el primer símbolo de la salida v.

El alfabeto de cinta Γ' de M incluye los alfabetos Σ y Γ, esto es, $\Sigma \cup \Gamma \subseteq \Gamma'$. También se dice que M *calcula o computa* la función f.

Para funciones numéricas se puede tomar $\Sigma = \{1\}$ y usar el sistema de numeración unitario: si n es un número natural, n se escribe como la secuencia de n unos 1^n, y la cadena vacía representa el número 0. Con esta representación el conjunto $\mathbb{N}$ de los números naturales se identifica con

$$\mathbb{N} = \{1^n : n \geq 0\} = \{\lambda, 1, 11, 111, 1111, 11111, \ldots\}$$

Como caso particular, una función numérica $f : \mathbb{N} \longrightarrow \mathbb{N}$ es Turing-computable si existe una MT estándar $M = (Q, q_0, q_a, \Sigma, \Gamma, \delta)$ tal que $q_0 1^m \vdash^* q_a 1^n$ siempre que $f(m) = n$, con $m, n \in \mathbb{N}$.

La noción de función Turing-computable se puede extender fácilmente a funciones de varios argumentos. Sea $(\Sigma^*)^k = \Sigma^* \times \cdots \times \Sigma^*$ (k veces). Una función $f : (\Sigma^*)^k \longrightarrow \Gamma^*$

parcial o total, con dominio $\mathrm{dom}(f)$, es Turing-computable si existe una MT estándar $M = (Q, q_0, q_a, \Sigma, \Gamma', \delta)$ tal que, para toda k-upla $(u_1, u_2, \ldots, u_k) \in (\Sigma^*)^k$ se tiene

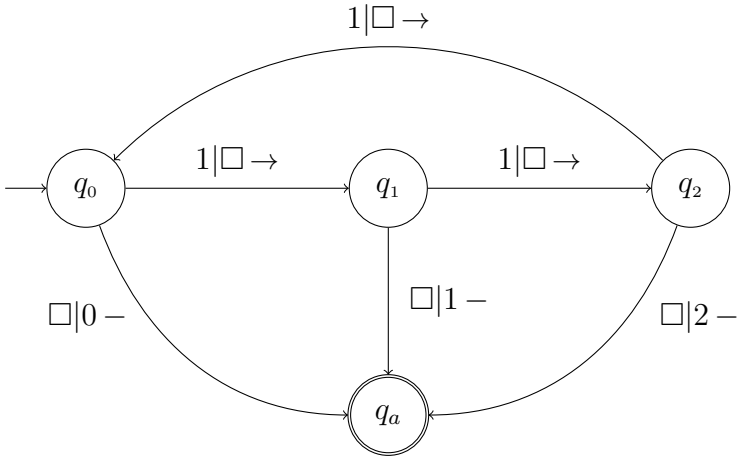
$$(u_1, u_2, \ldots, u_k) \in \mathrm{dom}(f) \implies q_0 u_1 \Box u_2 \Box \cdots \Box u_k \overset{*}{\vdash} q_a v, \quad \text{donde } v = f(u_1, u_2, \ldots, u_k).$$

Nótese que para escribir la entrada en la cinta, los k argumentos u_1, u_2,..., u_k están separados entre sí por una casilla en blanco. Igual que antes, no se permiten transiciones desde el estado final q_a.

Ejemplo Diseñar una MT M que calcule el residuo de división de n por 3, para cualquier número natural $n \geq 0$, escrito en el sistema de numeración unitario.

Solución. Aquí $\Sigma = \{1\}, \Gamma = \{0, 1, 2\}$ y f es la función total $f : \Sigma^* \longrightarrow \Gamma^*$ definida por $f(n) =$ residuo de división de n por 3, para todo $n \in \Sigma^*$.

Los posibles residuos de división por 3 son 0, 1 y 2, por lo cual bastan 3 estados, aparte de q_a, para calcular esta función. M recorre de izquierda a derecha la secuencia de entrada borrando los unos y pasando alternadamente por los estados q_0 (que representa el residuo 0), q_1 (residuo 1) y q_2 (residuo 2). El grafo de M se muestra a continuación.



Orden lexicográfico. Sea $\Sigma = \{s_1, s_2, \ldots, s_k\}$ un alfabeto dado en el cual los símbolos tienen un orden preestablecido, $s_1 < s_2 < \cdots < s_k$. En el conjunto Σ^* de todas las cadenas se define el *orden lexicográfico*, también denotado $<$, de la siguiente manera. Sean u, v dos cadenas en Σ^*,

$$u = a_1 a_2 \cdots a_m, \quad \text{donde } a_i \in \Sigma, \text{ para } 1 \leq i \leq m.$$
$$v = b_1 b_2 \cdots b_n, \quad \text{donde } b_i \in \Sigma, \text{ para } 1 \leq i \leq n.$$

Se tiene $u < v$ si

(1) $|u| < |v|$ (es decir, si $m < n$) o,

(2) $|u| = |v|$ (es decir, si $m = n$) y para algún índice i, $1 \leq i \leq m$, se cumple que

$$a_1 = b_1, a_2 = b_2, \ldots, a_{i-1} = b_{i-1}, a_i < b_i.$$

El orden lexicográfico entre cadenas es un orden lineal, o sea, para todo par de cadenas diferentes u, v en Σ^* se tiene $u < v$ o $v < u$, y es el orden utilizado para listar las palabras en los diccionarios de los lenguajes naturales.

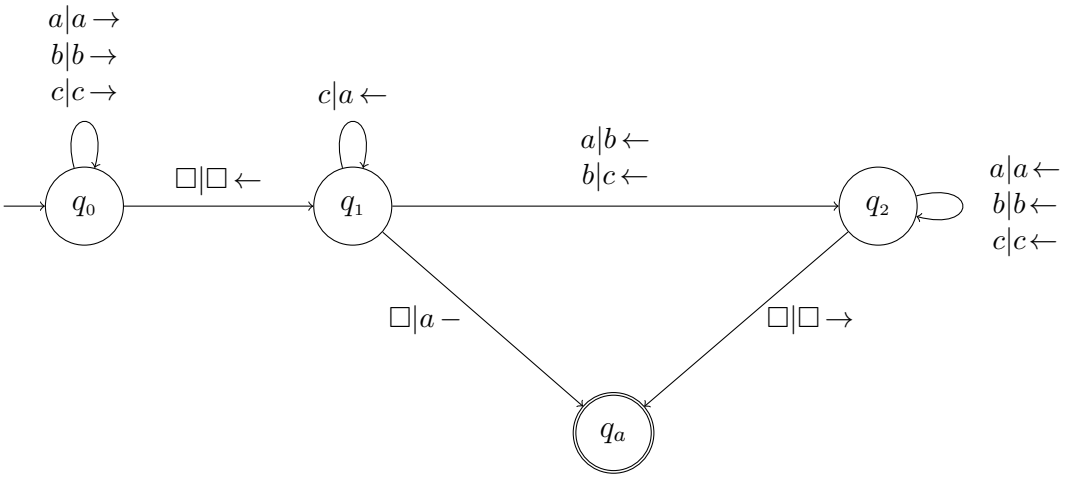
Ejemplo Sea $\Sigma = \{a, b, c\}$, donde los símbolos tienen el orden preestablecido $a < b < c$. La MT M exhibida abajo calcula la función $f : \Sigma^* \longrightarrow \Sigma^*$ definida por

$$f(u) = \text{cadena que sigue a } u \text{ en el orden lexicográfico.}$$

Las primeras cadenas de Σ^* ordenadas lexicográficamente son:

$$\lambda, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, aab, aac, aba, abb, abc, aca, acb, acc, baa, bab, \ldots$$

Hay que tener presente que la cadena que sigue a una cadena no-vacía u en el orden lexicográfico tiene la misma longitud que u, a menos que u solamente tenga $c$es. Así por ejemplo, $f(bab) = bac$, $f(babc) = baca$, $f(babccc) = bacaaa$, $f(cccc) = aaaaa$.



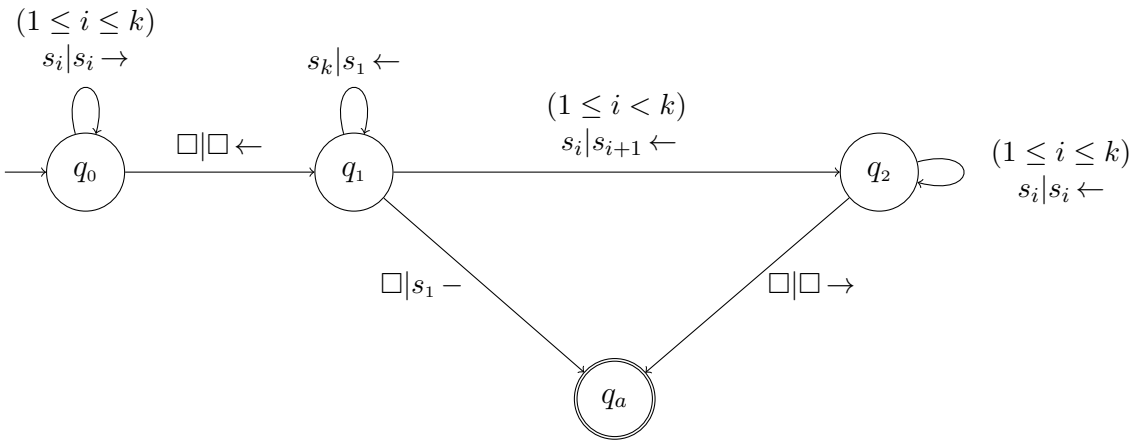
La MT M calcula $f(babccc) = bacaaa$ de la siguiente manera:

$$q_0babccc \overset{6}{\vdash} babcccq_0\square \vdash babcccq_1c \overset{3}{\vdash} baq_1baaa \vdash bq_2acaaa \overset{3}{\vdash} q_abacaaa.$$

Esta MT se puede modificar, sin necesidad de añadir más estados, para el caso de alfabetos con un número arbitrario de símbolos. Sea $\Sigma = \{s_1, s_2, \ldots, s_k\}$ un alfabeto dado en el cual los k símbolos tienen el orden preestablecido $s_1 < s_2 < \cdots < s_k$. La MT exhibida en la siguiente página calcula la función $f : \Sigma^* \longrightarrow \Sigma^*$ definida por

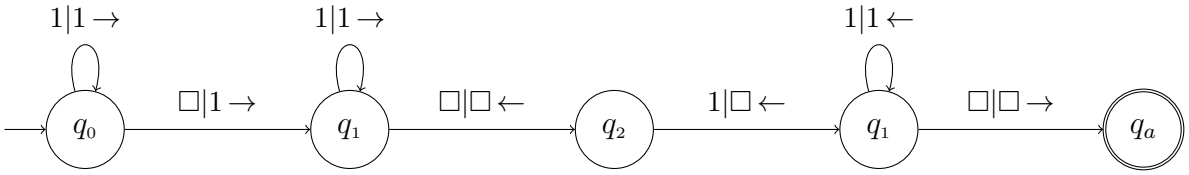
$$f(u) = \text{cadena que sigue a } u \text{ en el orden lexicográfico.}$$

Los bucles de los estados q_0 y q_2 tienen k instrucciones cada uno, con $1 \leq i \leq k$. Entre q_1 y q_2 hay $k - 1$ instrucciones, con $1 \leq i < k$.



Ejemplo Sea $\Sigma = \{1\}$. Diseñar una MT M que calcule la función parcial suma, en el sistema de numeración unitario, definida por $f(m, n) = m + n$, donde $n, m \geq 1$. Con entrada $1^m\square 1^n$, M debe producir como salida 1^{m+n}. Las secuencias de unos, 1^m y 1^n, representan los números naturales m y n, respectivamente.

Solución. La MT M que se exhibe a continuación transforma la entrada $1^m\square 1^n$ en la salida 1^{m+n} trasladando la cadena 1^n una casilla hacia la izquierda. Esto se consigue escribiendo un 1 en la casilla en blanco que hay entre 1^m y 1^n, y borrando el último 1 de 1^n.



Esta MT se puede modificar para calcular la función total suma, $f : \Sigma^* \longrightarrow \Sigma^*$ definida por $f(m, n) = m+n$, donde $m, n \geq 0$, es decir, incluyendo los casos $m = 0$ o $n = 0$ (véase el Ejercicio ① de la presente sección).

Ejercicios de la sección 6.4

Diseñar MT que calculen las siguientes funciones numéricas, utilizando para $\mathbb{N}$ el sistema de numeración unitario.

① $f : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$, donde $f(m, n) = m + n$.

② $f : \mathbb{N} \longrightarrow \mathbb{N}$, donde $f(n) = 2n$.

③ $f : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$, donde

$$f(m, n) = \begin{cases} 1, & \text{si } m \geq n, \\ 0, & \text{si } m < n. \end{cases}$$

④ $f : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$, donde $f(m, n) = \text{máx}(m, n)$.

⑤ $f : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$, donde $f(i, j, k) = j$. Esta función se llama "segunda proyección".

⑥ $f : \mathbb{N}^k \longrightarrow \mathbb{N}$ donde

$$f(m_1, \ldots, m_i, \ldots, m_k) = m_i.$$

f es la llamada i-ésima proyección.

⑦ $f : \mathbb{N} \longrightarrow \mathbb{N}$, donde $f(n) = 2^n$.

6.5. Máquinas de Turing no-deterministas (MTN)

El modelo no-determinista de máquina de Turing tiene los mismos componentes del modelo estándar, $M = (Q, q_0, q_a, \Sigma, \Gamma, \delta)$, con un único estado de aceptación q_a, pero se permite que, en un paso computacional, la unidad de control escoja aleatoriamente entre varias transiciones posibles. La función δ tiene la siguiente forma:

$$\delta(q, s) = \{(p_1, s_1, D_1), (p_2, s_2, D_2), \ldots, (p_k, s_k, D_k)\},$$

donde s y los s_i son símbolos pertenecientes a $\Gamma_\square$, los p_i son estados y cada D_i es un desplazamiento.

La noción de aceptación en una MTN es similar a la de los modelos no-deterministas de autómatas considerados antes: una cadena de entrada u es aceptada si existe por lo menos un procesamiento, a partir de la configuración inicial q_0u, que termine en una configuración de aceptación, vq_aw, con $v, w \in (\Gamma_\square)^*$.

Si $u \in \Sigma^*$, una secuencia de configuraciones instantáneas $C_0, C_1, \ldots, C_m$ tal que

$$C_0 \vdash C_1 \vdash \cdots \vdash C_m,$$

donde C_0 es la configuración inicial q_0u, se denomina una *computación* (de m pasos). Si C_m es una configuración de aceptación, la secuencia es llamada una *computación de aceptación de u*. El lenguaje aceptado por M se puede describir como

$$L(M) := \{u \in \Sigma^* : \text{existe una computación de aceptación } C_0 \vdash C_1 \vdash \cdots \vdash C_m\}.$$

El no determinismo no amplía la familia de lenguajes aceptados por máquinas de Turing; en otras palabras, una MTN se puede simular con una MT estándar, como se demuestra en el siguiente teorema.

6.5.1 Teorema. Todo lenguaje aceptado por una máquina de Turing no-determinista M puede ser aceptado por una MT estándar M'.

Demostración. Sea $M = (Q, q_0, q_a, \Sigma, \Gamma, \delta)$ una máquina de Turing no-determinista dada. La idea de la demostración es que si una cadena $u \in \Sigma^*$ es aceptada por M, la MT simuladora M' también va a aceptar a u después de realizar una búsqueda exhaustiva

entre todas las computaciones $C_0 \vdash C_1 \vdash \cdots \vdash C_m$, donde C_0 es la configuración inicial q_0u y $m \geq 1$. M' primero examina las computaciones de un paso, luego las computaciones de dos pasos, y así sucesivamente hasta encontrar una computación de aceptación.

Sea n el número máximo de transiciones en los conjuntos $\delta(q, s)$, considerando todo símbolo $s \in \Gamma_\square$ y todo estado $q \in Q$. Para cada $s \in \Gamma_\square$ y $q \in Q$, las transiciones contenidas en $\delta(q, s)$ se pueden enumerar entre 1 y n. Si hay menos de n transiciones en un $\delta(q, s)$ particular, se repite arbitrariamente una de ellas hasta completar n. De esta manera, cada una de las transiciones $\delta(q, s)$ se puede considerar como un conjunto ordenado, con exactamente n opciones indexadas:

$$\delta(q,s) = \{\underbrace{(p_1, b_1, D_1)}_{1}, \underbrace{(p_2, b_2, D_2)}_{2}, \ldots, \underbrace{(p_n, b_n, D_n)}_{n}\} \quad \leftarrow \text{índice}$$

En algunos conjuntos $\delta(q, s)$ habrá opciones repetidas, pero esta repetición no altera el lenguaje aceptado.

Sea $\Phi = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ un alfabeto con n símbolos nuevos (o sea, Γ y Φ son alfabetos disyuntos). Se puede pensar que el alfabeto $\{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ es una "codificación" del conjunto de números naturales $\{1, 2, \ldots, n\}$ (esto es, φ_1 representa a 1, φ_2 representa a 2 y así sucesivamente). La MT M' que simulará a M tendrá tres cintas. La primera cinta almacena la entrada, la segunda cinta contiene en todo momento una cadena de Φ^*, que representa una secuencia finita de números naturales entre 1 y n, y en la tercera cinta se hace la simulación de M propiamente dicha.

Concretamente, con entrada $u \in \Sigma^*$, M' realiza el siguiente procedimiento de 3 etapas:

(1) Copia la cadena de entrada u (que está escrita en la primera cinta) en la tercera cinta, y escribe la cadena $x = \varphi_1$ en la segunda cinta.

(2) Si $x = \varphi_{i_1}\varphi_{i_2}\cdots\varphi_{i_k}$ es la cadena escrita en la segunda cinta, M' simula en la cinta 3 la computación de k pasos que hace sobre u la máquina original M, utilizando en el paso j la opción con índice i_j del conjunto $\delta(q, s)$ correspondiente. Durante la simulación, el visor de la segunda cinta se va desplazando una casilla a la derecha de tal manera que en el j-ésimo paso está leyendo el símbolo φ_{i_j}. Si M' se detiene en algún momento en el estado de aceptación q_a, entonces M' acepta la entrada (como lo ha hecho M). Si después de estos k pasos la computación simulada no termina en aceptación, se pasa a la etapa (3).

(3) M' reemplaza la cadena x que está escrita en la segunda cinta por la cadena que sigue a x en el orden lexicográfico, sobre el alfabeto $\Phi = \{\varphi_1, \varphi_1, \ldots, \varphi_n\}$. Para ello, M' utiliza como subrutina la función Turing-computable presentada en la sección 6.4. A continuación, M' retorna a la etapa (2).

Si la MT original M acepta una cadena $u \in \Sigma^*$ siguiendo una computación de m pasos que corresponde a la sucesión $i_1, i_2, \ldots, i_m$ de enteros del conjunto $\{1, 2, \ldots, n\}$ (en el sentido

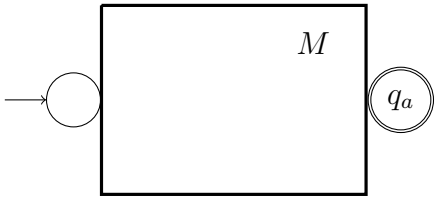
de que M utiliza en el paso r la opción i_r entre todas las opciones disponibles), entonces la MT M' simulará dicha computación cuando la cadena $x = \varphi_{i_1}\varphi_{i_2}\cdots\varphi_{i_m}$ esté escrita en la segunda cinta, y la cadena u será aceptada por M'. Esto demuestra que $L(M) \subseteq L(M')$. Recíprocamente, se tiene $L(M') \subseteq L(M)$ ya que M' solo puede aceptar cadenas que son aceptadas por M. Se concluye entonces que $L(M) = L(M')$.

Un detalle importante en la anterior simulación es que si la cadena $u \in \Sigma^*$ no es aceptada por la MT original M, entonces, con entrada u, la MT M' nunca se detendrá (entrará en un bucle infinito) porque regresará una y otra vez a la etapa (3), chequeando sin éxito todas las cadenas de Φ^*, en el orden lexicográfico. □

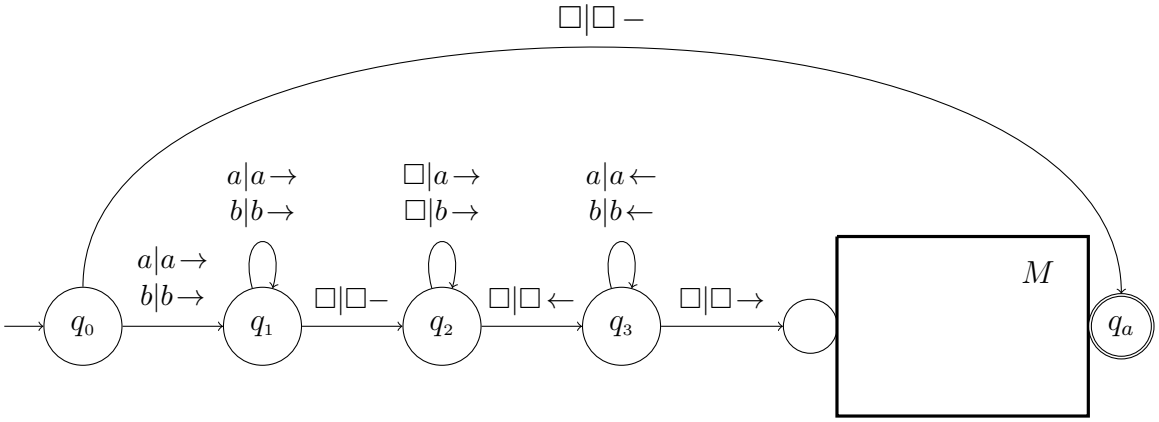
Ejemplo Para un lenguaje L sobre Σ se define P_L como el lenguaje de todos los prefijos de cadenas de L:

$$P_L = \{x \in \Sigma^* : (\exists y \in \Sigma^*)(xy \in L)\}.$$

Es claro que $\lambda \in P_L$ porque λ es prefijo de cualquier cadena, y $L \subseteq P_L$ porque toda cadena es prefijo de sí misma. Supóngase que el lenguaje L es Turing-aceptable; existe entonces una MT M, modelo estándar, con un único estado de aceptación q_a, tal que $L(M) = L$:



Una MTN M' tal que $L(M') = P_L$ se muestra en la siguiente gráfica.



M' actúa así: con una entrada $x \in \Sigma^*$ sobre la cinta, su cabeza se mueve, en el estado q_1, hacia el extremo derecho de x y escribe aleatoriamente una cadena $y \in \Sigma^*$, en el estado q_2. Las transiciones que salen de q_2 constituyen la parte no-determinista del funcionamiento de M'. A continuación, M' retorna al extremo izquierdo, en el estado q_3, y procesa la

cadena xy escrita en cinta utilizando la máquina original M. Si realmente $x \in P_L$, entre todos los cómputos posibles con entrada x, M' encontrará eventualmente una cadena $y \in \Sigma^*$ tal que $xy \in L$ y aceptará a x. Si $x \notin P_L$, ninguna de las elecciones de y conducirá a la aceptación.

La MTN M' está diseñada para implementar el siguiente algoritmo no-determinista:

Algoritmo. Con entrada $x \in \Sigma^*$:

(1) Escribir aleatoriamente una cadena y perteneciente a Σ^*.

(2) Correr la MT M con entrada xy. Si M acepta la cadena xy, aceptar la entrada x.

La parte (1) es la etapa no-determinista del algoritmo y constituye una "conjetura". La parte (2) es una etapa de "verificación", que es completamente determinista. Muchos algoritmos no-deterministas tienen esta estructura de dos etapas que se puede sintetizar en la fórmula

$$\text{NO-DETERMINISMO} = \text{CONJETURA} + \text{VERIFICACIÓN}.$$

Según el Teorema 6.5.1 para la MTN M' se puede construir una MT determinista M'' tal que $L(M') = L(M'') = P_L$. Con entrada x, la máquina simuladora M'' reemplaza la etapa de conjetura por la búsqueda exhaustiva de un sufijo y tal que $xy \in L$. Si la cadena x no está en P_L, esta búsqueda es infructuosa y la máquina M'' nunca se detendrá con entrada x.

Una máquina de Turing multicinta no-determinista se define como el modelo multicinta de la sección 6.3.3 pero permitiendo un número finito de opciones en cada transición. Utilizando un argumento similar al del Teorema 6.5.1, se puede demostrar que este modelo es equivalente al modelo estándar. Este hecho se enuncia en el siguiente teorema.

6.5.2 Teorema. Todo lenguaje aceptado por una máquina de Turing multicinta no-determinista puede ser aceptado por una máquina de Turing estándar.

Ejercicios de la sección 6.5

① Sea L un lenguaje sobre Σ y M una MT, modelo estándar, tal que $L(M) = L$. Construir MTN (Máquinas de Turing no-deterministas) que acepten los siguientes lenguajes:

(i) El lenguaje S_L de todos los sufijos de cadenas de L; es decir,

$$S_L = \{x \in \Sigma^* : (\exists y \in \Sigma^*)(yx \in L)\}.$$

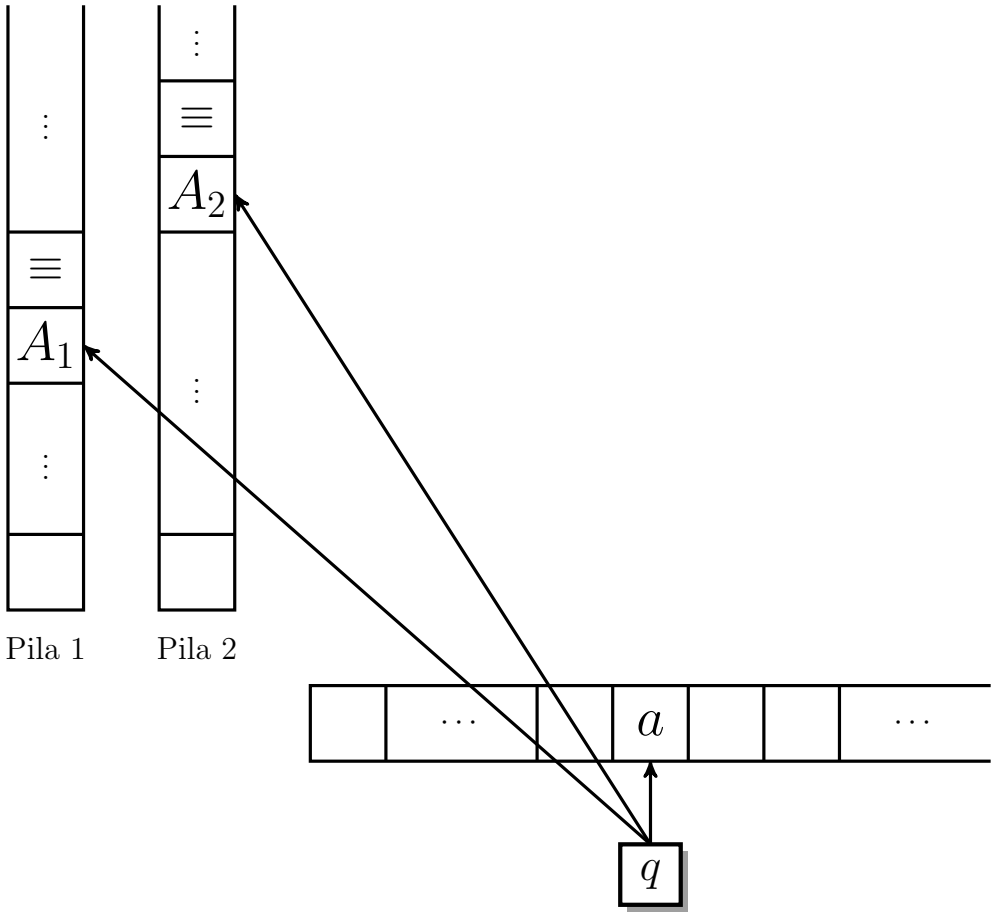
(ii) El lenguaje C_L de todas las subcadenas de cadenas de L; es decir,

$$C_L = \{x \in \Sigma^* : (\exists y, z \in \Sigma^*)(yxz \in L)\}.$$

② Sea $\Sigma = \{a, b\}$ y $L = \{ww : w \in \Sigma^*\}$. Diseñar una Máquina de Turing no-determinista (con una o varias cintas) que acepte el lenguaje L.

6.6. Autómatas con dos pilas (AF2P)

En esta sección presentaremos el modelo de autómata con dos pilas (AF2P) que resulta también equivalente al modelo estándar de máquina de Turing. Un autómata con dos pilas es esencialmente un AFPN, tal como se definió en el capítulo 4, con la adición de una pila más. Podríamos también definir autómatas con k pilas, $k \geq 3$, pero tal modelo no aumenta la capacidad computacional que se consigue con dos pilas. Las pilas tienen la misma restricción que antes: el autómata sólo tiene acceso al símbolo que está en el tope de cada pila. Un paso computacional depende del estado actual de la unidad de control, del símbolo escaneado en la cinta y de los dos topes de pila, como se muestra en la siguiente gráfica.

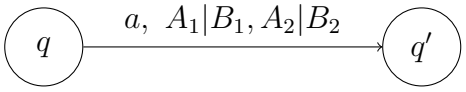


Al procesar un símbolo $a \in \Sigma$, la unidad de control se desplaza una casilla a la derecha, cambia al estado q' (que puede ser el mismo q) y realiza sobre cada pila una de las siguientes cuatro acciones: o reemplaza el tope de la pila por otro símbolo, o añade un nuevo símbolo a la pila, o borra el tope de la pila, o no altera la pila. Estas acciones permitidas en las pilas son las mismas que en el modelo AFPN y están definidas en términos de la función de transición Δ en la que las instrucciones básicas tienen la forma

$$\Delta(q, a, A_1, A_2) = (q', B_1, B_2).$$

Se tienen en cuenta todos los casos $a \in \Sigma \cup \{\lambda\}$ y $A_1, A_2, B_1, B_2 \in \Gamma \cup \{\lambda\}$. El caso $a = \lambda$ corresponde a las transiciones λ.

La instrucción $\Delta(q, a, A_1, A_2) = (q', B_1, B_2)$ se representa en el grafo de estados en la siguiente forma:



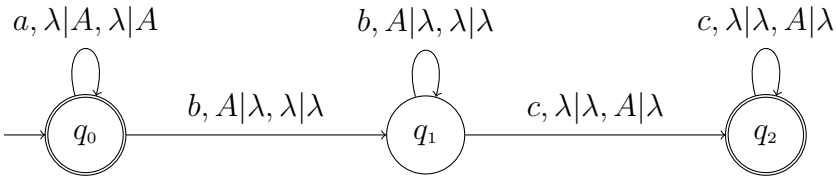
El modelo AF2P es no-determinista, de manera que cada $\Delta(q, a, A_1, A_2)$ es, en general, un conjunto finito de opciones, cada una de las cuales es una instrucción básica. Inicialmente las dos pilas están vacías y la unidad de control está escaneando el fondo de cada pila. La cadena de entrada se coloca en la cinta de entrada, en la forma usual. Nótese que sólo es necesario exigir que la cinta de entrada sea infinita en una dirección ya que la unidad de control no puede nunca retornar a la izquierda.

Una configuración instantánea es una cuádrupla de la forma $[q, v, \beta_1, \beta_2]$ que representa lo siguiente: la unidad de control está en el estado q, v es la parte no procesada de la cadena de entrada (la unidad de control está escaneando el primer símbolo de v), y las cadenas β_1 y β_2 son el contenido total de la pila 1 y de la pila 2, respectivamente. Tanto β_1 como β_2 se leen en la pila de arriba hacia abajo, así que el tope de la pila i es el primer símbolo de β_i, $i = 1, 2$. Si $w \in \Sigma^*$ es una entrada, la configuración inicial es $[q_0, w, \lambda, \lambda]$ y una configuración de aceptación es de la forma $[p, \lambda, \lambda, \lambda]$ donde $p \in F$; es decir, para ser aceptada, una entrada debe ser procesada completamente, el procesamiento debe terminar con las dos pilas vacías y la unidad de control en un estado de aceptación. El lenguaje aceptado por un AF2P M se define entonces como:

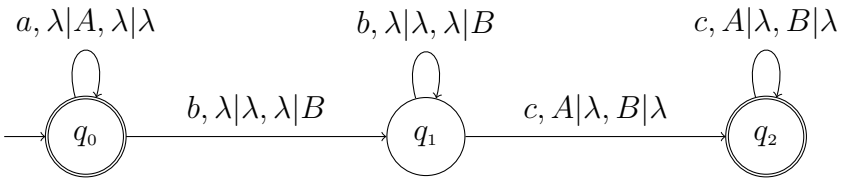
$$L(M) := \{w \in \Sigma^* : \text{existe un procesamiento } [q_0, w, \lambda, \lambda] \stackrel{*}{\vdash} [p, \lambda, \lambda, \lambda],\ p \in F\}.$$

Ejemplo Diseñar un AF2P que acepte el lenguaje $L = \{a^n b^n c^n : n \geq 0\}$.

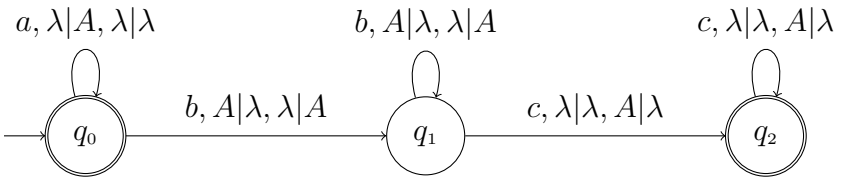
Solución 1. Un plan natural es apilar las $a$es en ambas pilas. Al aparecer las $b$es se va vaciando la primera pila (sin alterar la segunda) y al aparecer las $c$es se va vaciando la segunda pila (sin alterar la primera). Si la entrada tiene la forma $a^n b^n c^n$, se consume completamente la entrada y se vacían ambas pilas. Por el contrario, si la entrada no es de la forma $a^n b^n c^n$, o bien no se consume toda la entrada, o bien no desocupa alguna de las pilas. Para implementar esta idea utilizamos tres estados, siendo q_2 el único estado de aceptación. El estado inicial q_0 debe ser de aceptación para aceptar la cadena λ. El alfabeto de pila es $\Gamma = \{A\}$.



Solución 2. También es posible apilar las $a$es (como Aes) en la primera pila y las $b$es (como Bes) en la segunda pila, y luego vaciar simultáneamente ambas pilas cuando aparecen las $c$es en cinta. En este caso, el alfabeto de pila es $\Gamma = \{A, B\}$.



Solución 3. Otra manera de aceptar a L es apilar las $a$es en la primera pila únicamente, dejando la segunda pila vacía. Al aparecer las $b$es, se va vaciando la primera pila y se inserta en la segunda pila una A por cada b leída. Finalmente, al aparecer las $c$es se va vaciando la segunda pila sin alterar la primera. El alfabeto de pila es $\Gamma = \{A\}$.



Ejercicios de la sección 6.6

① Sea $\Sigma = \{a, b, c\}$. Diseñar AF2P (Autómatas Finitos con dos pilas) que acepten los siguientes lenguajes En cada caso, explicar brevemente el plan utilizado en el diseño y presentar el autómata por medio de un grafo de estados.

(i) $L = \{a^n b^{2n} c^n : n \geq 0\}$.

(ii) $L = \{a^k b^m c^n : 0 \leq k \leq m \leq n\}$.

(iii) $L = \{a^k b^m c^n : k \geq m \geq n \geq 0\}$.

(iv) $L = \{a^m b^n c^n : m, n \geq 1,\ m > n\}$.

(v) $L = \{a^m b^n c^n : m, n \geq 1,\ n > m\}$.

(vi) $L = \{a^m b^n a^m c^n : m \neq n\}$.

(vii) $L = \{a^m b^n a^m c^n : m, n \geq 1\}$.

(viii) $L = \{u \in \Sigma^* : \#_a(u) = \#_b(u) = \#_c(u)\}$.

② Sea $\Sigma = \{a, b\}$ y $L = \{ww : w \in \Sigma^*\}$. Diseñar un AF2P (Autómata Finito con dos pilas) que acepte el lenguaje L.

③ Sea $\Sigma = \{a, b, c, d\}$ y $L = \{a^n b^n c^n d^n : n \geq 0\}$. Diseñar un AF2P (Autómata Finito con dos pilas) que acepte el lenguaje L.

6.7. La tesis de Church-Turing

Si bien la máquina de Turing antecedió en varias décadas a la implementación física de los computadores actuales, ha resultado ser un modelo muy conveniente para representar "lo computable" (lo que es capaz de hacer *cualquier* dispositivo físico de computación secuencial). La declaración conocida como "tesis de Church-Turing" afirma precisamente que la Máquina de Turing es un modelo general de computación secuencial: cualquier procedimiento algorítmico que pueda realizar un ser humano o un computador digital se puede llevar a cabo mediante una Máquina de Turing. Se establece así una conexión intuitiva directa entre máquinas de Turing y algoritmos.

6.7.1. ***Tesis de Church-Turing****. Todo algoritmo puede ser descrito por medio de una máquina de Turing.*

En su formulación más amplia, la tesis de Church-Turing abarca tanto los algoritmos que producen una salida para cada entrada como aquéllos que no terminan (ingresan en bucles infinitos) para algunas entradas.

Para apreciar su significado y su alcance, hay que enfatizar que la Tesis de Church-Turing no es un enunciado matemático susceptible de demostración, ya que involucra la noción intuitiva (no definida rigurosamente) de *algoritmo*. Por tal razón la tesis no se puede demostrar como un teorema pero sí se podría refutar exhibiendo un procedimiento efectivo, que todo el mundo acepte que es un verdadero algoritmo y que no pueda ser descrito por medio de una máquina de Turing. Pero tal refutación no se ha producido hasta la fecha; de hecho, la experiencia acumulada durante décadas de investigación ha corroborado una y otra vez la tesis de Church-Turing.

Hay dos hechos más que contribuyen a apoyar la tesis:

1. La adición de recursos computacionales a las máquinas de Turing (múltiples pistas o cintas, no determinismo, etc) no incrementa el poder computacional del modelo básico. Esto es un indicativo de que la máquina de Turing no sólo es un modelo extremadamente robusto sino que representa el límite de lo que un dispositivo de computación secuencial puede hacer.

2. Todos los modelos o mecanismos computacionales propuestos para describir formalmente la noción de algoritmo han resultado ser equivalentes a la máquina de Turing, en el sentido de que lo que se puede hacer con ellos también se puede hacer con una MT adecuada, y viceversa. Entre los modelos de computación equivalentes a la máquina de Turing podemos citar:

   - Las funciones parciales recursivas (modelo de Gödel y Kleene, 1936).
   - El cálculo-λ (modelo de Church, 1936).
   - Sistemas de deducción canónica (modelo de Post, 1943).
   - Lógica combinatoria (modelos de Schönfinkel, 1924; Curry, 1929)

- Algoritmos de Markov (modelo de Markov, 1951).
- Las gramáticas irrestrictas (modelo de Chomsky, 1956).
- Las máquinas de registro (modelo de Shepherdson-Sturgis, 1963).

La equivalencia mutua de todos los modelos propuestos refuerza la idea que ellos capturan la esencia de lo que significa "computación efectiva".

6.8. Codificación de Máquinas de Turing

Toda MT se puede codificar como una secuencia binaria finita, es decir, una secuencia finita de ceros y unos. En esta sección presentaremos un esquema de codificación para todas las MT (modelo estándar) que actúen sobre un alfabeto de entrada $\Sigma = \{s_2, s_3, \ldots, s_m\}$ predeterminado. Para definir una máquina de Turing se utilizan los siguientes símbolos:

(1) Estados disponibles: q_1, q_2, q_3, q_4, ...
q_1 es siempre el estado inicial y q_2 es el único estado de aceptación.

(2) Símbolos de cinta disponibles: $s_1, s_2, s_3, \ldots, s_m, s_{m+1}, s_{m+2}, \ldots, \ldots$
Aquí, $s_1 = \square$ (símbolo que representa una casilla en blanco), $\Sigma = \{s_2, s_3, \ldots, s_m\}$ es el alfabeto de entrada común (fijo) de todas las MT, y s_k, con $k \geq m+1$, son los símbolos de cinta disponibles (cada MT utiliza su propia colección finita de símbolos auxiliares).

Cada MT posee únicamente un número finito de estados y un número finito de símbolos de cinta, pero hay una secuencia infinita de símbolos de ambos tipos disponibles para todas las MT.

Una instrucción determinada I en una MT, $\delta(q_i, s_j) = (q_k, s_\ell, D_t)$, se codifica por medio de la cadena binaria

$$01^i01^j01^k01^\ell01^t0.$$

Es decir, los estados y los símbolos de cinta se codifican como secuencias de unos, utilizando ceros como separadores. La codificación de una instrucción utiliza exactamente seis ceros separados por secuencias de unos. El exponente final t asume uno de los siguientes valores: 1 (desplazamiento a la derecha, $\rightarrow$), 2 (desplazamiento a la izquierda, $\leftarrow$) o 3 (estacionario, $-$). La codificación de una instrucción I se denota como $\langle I \rangle$.

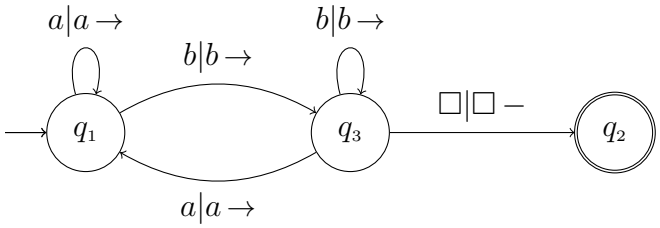
Una MT M está definida completamente por su función de transición, es decir, por su lista de instrucciones: $\{I_1, I_2, \ldots, I_r\}$ (un número finito de instrucciones). Una codificación de M, denotada $\langle M \rangle$, se obtiene concatenando las codificaciones de todas sus instrucciones (o transiciones). Más precisamente,

$$\langle M \rangle = \langle I_1 \rangle \langle I_2 \rangle \cdots \langle I_r \rangle 00.$$

En una codificación de M las instrucciones aparecen separadas por 00 y $\langle M \rangle$ termina en 000. Puesto que el orden en que se presentan las transiciones de una MT no es relevante,

una misma MT tiene varias codificaciones diferentes. De hecho, como se puede cambiar arbitrariamente el nombre de los estados (excepto q_1 y q_2), toda máquina de Turing con tres o más estados tiene infinitas codificaciones. Esto no representa ninguna desventaja práctica ni conceptual ya que no se pretende que las codificaciones sean únicas.

Ejemplo Sea $\Sigma = \{a, b\}$ y M la siguiente MT que acepta el lenguaje de todas las cadenas que terminan en b:



Como el alfabeto de entrada es $\Sigma = \{a, b\}$, entonces $s_1 = \square$, $s_2 = a$ y $s_3 = b$. Las cinco instrucciones de M (en orden arbitrario) y sus respectivas codificaciones son:

I_1 :	$\delta(q_1, s_2) = (q_1, s_2, \rightarrow)$.	Codificación de I_1:	0101101011010.
I_2 :	$\delta(q_1, s_3) = (q_3, s_3, \rightarrow)$.	Codificación de I_2:	01011101110111010.
I_3 :	$\delta(q_3, s_2) = (q_1, s_2, \rightarrow)$.	Codificación de I_3:	011101101011010.
I_4 :	$\delta(q_3, s_3) = (q_3, s_3, \rightarrow)$.	Codificación de I_4:	0111011101110111010.
I_5 :	$\delta(q_3, s_1) = (q_2, s_1, -)$.	Codificación de I_5:	0111010110101110.

Manteniendo este orden, una codificación de M sería $\langle M\rangle = \langle I_1\rangle\langle I_2\rangle\langle I_3\rangle\langle I_4\rangle\langle I_5\rangle 00$. Explícitamente,

$$\langle M\rangle = 0101101011010010111011101110100111011010110100111011101110111010011101011010111000,$$

la cual se puede escribir también como

$$0101^2 0101^2 0100101^3 01^3 01^3 01001^3 01^2 0101^2 01001^3 01^3 01^3 01^3 01001^3 0101^2 0101^3 000.$$

Cambiando el orden de las cinco transiciones de M obtendríamos en total $5! = 120$ codificaciones diferentes para M.

Es claro que no todas las secuencias binarias representan máquinas de Turing: la codificación de una MT no puede comenzar con 1 ni pueden aparecer cuatro ceros consecutivos. Así, las secuencias 01010000110, 0100001110 y 1011010110111010 no codifican ninguna MT. Una cadena binaria que codifique una MT se denomina un *código válido (o una codificación válida) de una MT*. Podemos concebir un algoritmo que determine si una secuencia binaria finita codifica o no una MT.

Algoritmo para verificar si una cadena binaria es un código válido de una MT. Una cadena en $u \in \{0,1\}^*$ es un código válido de una MT si

(1) $u \in (01^+01^+01^+01^+01^+0)^+00$. Esta expresión regular representa las concatenaciones de bloques con exactamente seis ceros que separan secuencias de unos. Cada uno de esos bloques contiene cadenas de la forma $01^i01^j01^k01^\ell01^t0$, que representan instrucciones individuales. La cadena u debe entonces terminar en 000 (los dos ceros finales requeridos más el cero proveniente de la última instrucción).

(2) En cada bloque (instrucción) $01^i01^j01^k01^\ell01^t0$, $t \in \{1,2,3\}$. Es decir, la unidad de control solo tiene tres posibles desplazamientos (derecha, izquierda, estacionario).

(3) No hay dos bloques (instrucciones) que comiencen con la misma subcadena 01^i01^j0. Es decir, la máquina de Turing es determinista.

(4) No hay ningún bloque (instrucción) que comience con 01^20. Es decir, no hay transiciones desde el estado de aceptación.

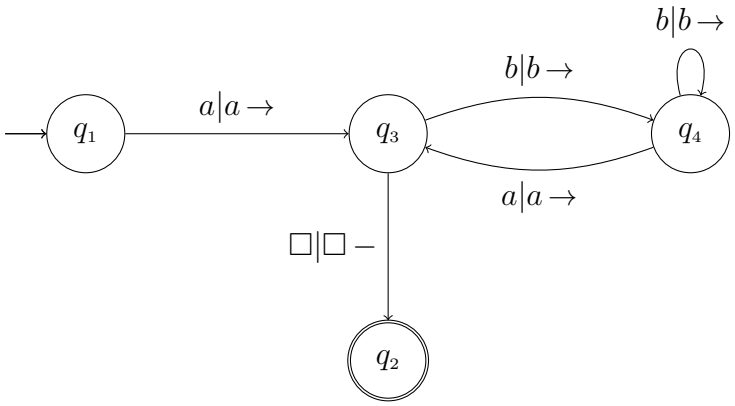
Una MT puede tener muchas codificaciones pero un código válido se puede decodificar y representa una única MT. En otras palabras, la codificación no es única pero la decodificación sí lo es. Si una cadena binaria no es un código válido de una MT, supondremos que codifica una MT sin transiciones que no acepta ninguna cadena, es decir, acepta el lenguaje $\varnothing = \{\ \}$. De esta forma, *todas las cadenas binarias representan máquinas de Turing*. Puesto que $\{0,1\}^*$ se puede ordenar lexicográficamente, podemos entonces hablar de la i-ésima máquina de Turing, la cual denotaremos por M_i. Nótese que en la enumeración M_1, M_2, M_3, ..., cada MT aparece varias veces (porque al cambiar el orden de las transiciones se obtiene una codificación diferente). Las MT que aceptan el lenguaje $\varnothing$ aparecen infinitas veces en la enumeración.

La enumeración M_1, M_2, M_3,... de todas las máquinas de Turing sobre un alfabeto de entrada Σ induce una enumeración de todos los lenguajes Turing-aceptables, a saber, $L_1 = L(M_1)$, $L_2 = L(M_2)$, $L_3 = L(M_3)$, ... Como conclusión podemos enunciar el siguiente teorema.

6.8.1 Teorema. Dado Σ, el conjunto de todas las máquinas de Turing con alfabeto de entrada Σ es infinito enumerable. También lo es el conjunto de todos los lenguajes Turing-aceptables.

Ejercicios de la sección 6.8

① Sea M la siguiente MT cuyo alfabeto de entrada es $\Sigma = \{a,b\}$:



Determinar el lenguaje aceptado por M y codificar la máquina M siguiendo el esquema presentado en la presente sección ($s_1 = \square$, $s_2 = a$ y $s_3 = b$).

② Las siguientes cadenas binarias son códigos válidos de MT que actúan sobre el alfabeto de entrada $\Sigma = \{a, b\}$, siguiendo el esquema de codificación presentado en la presente sección ($s_1 = \square$, $s_2 = a$ y $s_3 = b$). Decodificar las máquinas y determinar en cada caso el lenguaje aceptado.

(i) $0101101110110100111011101111011101100111101110 \xrightarrow{\text{sigue}}$
$1111011101001111011011110110100111101011010111000.$

(ii) $0101^3 0101^3 0100101^2 01^3 01^2 01001^4 01^2 01^5 01^2 01001^3 01^2 01^4 01^2 010 \xrightarrow{\text{sigue}}$
$01^4 01^3 0101^3 01001^5 0101^2 0101^3 000.$

③ Escribir las codificaciones de las siguientes MT: M_7, M_{14}, M_{65}, M_{150} y M_{255}. ¿Cuál es el lenguaje aceptado por tales máquinas?

6.9. Lenguajes que no son Turing-aceptables

Utilizando argumentos de cardinalidad, es posible concluir rápidamente que existen lenguajes que no son Turing-aceptables. Dado un alfabeto Σ, el conjunto de todos los subconjuntos de Σ^*, es decir, $\wp(\Sigma^*)$, es el universo de todos los lenguajes sobre Σ. Por el llamado Teorema de Cantor, $\wp(\Sigma^*)$ no es enumerable, pero según el Teorema 6.8.1, el conjunto de todos los lenguajes Turing-aceptables sí es enumerable. Por consiguiente, existen infinitos lenguajes que no pueden ser aceptados por ninguna máquina de Turing.

En esta sección exhibiremos un lenguaje concreto que no es Turing-aceptable. Recordemos que para concluir que ciertos lenguajes no son regulares utilizamos razonamientos por contradicción. Aquí también razonaremos por contradicción, aunque el lenguaje definido es completamente artificial. Dado un alfabeto de entrada $\Sigma = \{s_2, \ldots, s_m\}$, las cadenas de Σ^* se pueden ordenar lexicográficamente estableciendo primero un orden arbitrario (pero fijo) para los símbolos de Σ, por ejemplo, $s_2 < s_3 < \cdots < s_m$. Obtenemos así una enumeración w_1, w_2, w_3, ... de todas las cadenas de Σ^*. Esto nos permite hablar de la

i-ésima cadena de entrada, w_i. La interacción entre estas dos enumeraciones diferentes w_1, w_2, w_3, ... (entradas) y M_1, M_2, M_3, ... (máquinas de Turing) produce un lenguaje que no es Turing-aceptable.

6.9.1 Teorema. Sea Σ un alfabeto dado. El lenguaje

$$L = \{w_i \in \Sigma^* : w_i \text{ no es aceptada por } M_i\}$$

no es Turing-aceptable.

Demostración. Razonamos por contradicción (o reducción al absurdo). Si L fuera Turing-aceptable, existiría una MT M_k (para algún $k \geq 1$), con respecto a la enumeración de máquinas de Turing descrita en la sección 6.8, tal que $L = L(M_k)$. Se tendría entonces

$$w_k \in L \Longrightarrow w_k \text{ no es aceptada por } M_k \Longrightarrow w_k \notin L(M_k) = L.$$
$$w_k \notin L \Longrightarrow w_k \text{ es aceptada por } M_k \Longrightarrow w_k \in L(M_k) \Longrightarrow w_k \in L.$$

Por lo tanto, $w_k \in L \Longleftrightarrow w_k \notin L$, lo cual es una contradicción. □

El lenguaje L del Teorema 6.9.1 se denota como L_d y se denomina *lenguaje de diagonalización*; esta terminología proviene del hecho de que la demostración contiene un "argumento diagonal" que se puede visualizar de la siguiente manera. En la siguiente matriz infinita hay un 1 en la posición (i, j) si M_i acepta a w_j, y un 0 si M_i no acepta a w_j:

	w_1	w_2	w_3	w_4	$\cdots$
M_1	0	0	1	0	$\cdots$
M_2	1	0	0	0	$\cdots$
M_3	0	1	1	0	$\cdots$
$\vdots$		$\vdots$			

Las filas de la matriz representan *todos* los lenguajes Turing-aceptables. El lenguaje L_d se construye "complementando" la diagonal (cambiando ceros por unos y unos por ceros). Tal lenguaje no puede estar en esa lista y, por lo tanto, no puede ser Turing-aceptable. Se trata del mismo argumento utilizado para demostrar que el conjunto de todas las sucesiones infinitas de ceros y unos no es enumerable.

6.10. Máquina de Turing universal

En la sección 6.8 se presentó un esquema de codificación para todas las máquinas de Turing con un alfabeto de entrada predeterminado $\Sigma = \{s_2, s_3, \ldots, s_m\}$ (El símbolo s_1 ha sido reservado para □). Es necesario ahora codificar todas las cadenas de Σ^* para lo cual utilizaremos un esquema muy simple, compatible con la codificación de MT: el símbolo s_i se codifica como 1^i y una cadena $s_{i_1}s_{i_2}\cdots s_{i_k}$ se codifica como $1^{i_1}01^{i_2}0\cdots01^{i_k}0$.

Ejemplo Sea $\Sigma = \{a, b, c, d\}$. Podemos asignar los símbolos así: $s_2 = a$, $s_3 = b$, $s_4 = c$, y $s_5 = d$. La cadena *dbabc* se codifica como $1^5 01^3 01^2 01^3 01^4 0$.

Una codificación válida de una cadena en Σ^* debe constar de bloques de unos separados por ceros individuales, es decir, deber ser de la forma $1^{i_1}01^{i_2}0\cdots01^{i_k}0$, con la restricción $2 \leq i_j \leq m$ para todo exponente i_j. En otras palabras, una cadena binaria u es una codificación válida si $u \in (11^+0)^+$. A diferencia de la codificación de máquinas de Turing, la codificación de cadenas en este esquema es única (una vez que a cada símbolo de Σ le sea asignado un símbolo s_i). La codificación de una cadena $w \in \Sigma^*$ se denota como $\langle w \rangle$.

Una máquina de Turing universal $M_{\mathcal{U}}$ simula el funcionamiento de todas las MT (sobre el alfabeto de entrada Σ). $M_{\mathcal{U}}$ está diseñada para procesar cadenas binarias de la forma $\langle M \rangle \langle w \rangle$, siendo M una MT determinada y w una cadena de entrada perteneciente a Σ^*.

$M_{\mathcal{U}}$ es una MT con tres cintas cuyo alfabeto de cinta es $\{0, 1\}$. $M_{\mathcal{U}}$ utiliza el símbolo externo $\square$ (que representa una casilla en blanco) como se hace en el modelo estándar. Hay que aclarar que las demás máquinas de Turing utilizan el símbolo blanco codificado como 1, por la asignación previa $s_1 = \square$. Las entradas que lee $M_{\mathcal{U}}$ son cadenas binarias que se escriben en la primera cinta (cinta de entrada); las otras dos cintas están inicialmente en blanco. $M_{\mathcal{U}}$ no escribe nunca sobre la primera cinta (que es entonces una cinta de lectura).

A continuación se detalla el funcionamiento de la máquina $M_{\mathcal{U}}$, invocando apropiadamente la tesis de Church-Turing:

1. Copia la entrada en la cinta 2 y verifica que sea de la forma $\langle M \rangle \langle w \rangle$. Puesto que el código de una MT M termina en 000, en la cadena $\langle M \rangle \langle w \rangle$ aparecen tres ceros consecutivos únicamente en el sitio que separa los códigos de M y w. Chequea que $\langle M \rangle$ sea una codificación válida de una MT, y $\langle w \rangle$ sea una codificación válida de una cadena de Σ^*. Si la entrada no es de la forma $\langle M \rangle \langle w \rangle$, $M_{\mathcal{U}}$ se detiene sin aceptar.

2. Si la entrada es de la forma $\langle M \rangle \langle w \rangle$, $M_{\mathcal{U}}$ copia el código de w en la segunda cinta. Como se indicó antes, los códigos de M y w están separados por tres ceros consecutivos. $M_{\mathcal{U}}$ utilizará la segunda cinta para simular el procesamiento que hace M de w.

3. La cadena 1, que representa el estado inicial q_1, se coloca en la tercera cinta. La unidad de control pasa a escanear el primer símbolo de cada cadena binaria, en cada una de las tres cintas. La tercera cinta se usa para almacenar el estado actual de M, también codificado: q_1 se codifica como 1, q_2 se codifica como 11, q_3 como 111, etc.

4. $M_{\mathcal{U}}$ utiliza la información de las cintas 2 y 3 para buscar en la cinta 1 la transición que sea aplicable. Si encuentra una transición aplicable, $M_{\mathcal{U}}$ simula en la cinta 2 lo que haría M y cambia el estado señalado en la cinta 3. Esto requiere re-escribir la cadena de la cinta 2 desplazando adecuadamente los símbolos a izquierda o a derecha. La simulación continúa de esta forma, si hay transiciones aplicables. Después de realizar una transición, la unidad de control regresa, en la primera y tercera cintas, al primer símbolo de la cadena.

Si al procesar una entrada w, $M_{\mathcal{U}}$ se detiene en el único estado de aceptación de M (que es q_2 y está codificado como 11), entonces la cadena w será aceptada. Por consiguiente, $M_{\mathcal{U}}$ tiene también un único estado de aceptación, q_2, que es el mismo estado de aceptación de cualquier otra MT.

Si durante la simulación $M_{\mathcal{U}}$ no encuentra una transición aplicable o se detiene en un estado que no es de aceptación, se detiene sin aceptar, como lo haría M.

Se tiene entonces que $M_{\mathcal{U}}$ acepta la entrada $\langle M\rangle\langle w\rangle$ si y solamente si M acepta a w. De modo que el lenguaje aceptado por la máquina de Turing universal $M_{\mathcal{U}}$ se puede describir explícitamente; este lenguaje se denomina corrientemente el *lenguaje universal* y se denota con $L_{\mathcal{U}}$:

$$L_{\mathcal{U}} = \{\langle M\rangle\langle w\rangle : \text{la MT } M \text{ acepta la cadena } w \in \Sigma^*\}.$$

El lenguaje universal $L_{\mathcal{U}}$ es, por consiguiente, un lenguaje Turing-aceptable. Sin embargo, este lenguaje no es Turing-decidible, como se establece en el siguiente teorema.

6.10.1 Teorema. El lenguaje universal, $L_{\mathcal{U}} = \{\langle M\rangle\langle w\rangle : M \text{ acepta a } w\}$, es Turing-aceptable pero no es Turing-decidible.

Demostración. Para demostrar que $L_{\mathcal{U}}$ no es Turing-decidible razonamos por contradicción: suponemos que existe una MT $\mathcal{M}$ que procesa todas las entradas $\langle M\rangle\langle w\rangle$ y se detiene siempre en un estado de aceptación (si M acepta a w) o en uno de rechazo (si M no acepta a w). Esta suposición permitirá construir una MT $\mathcal{M}'$ que acepte el lenguaje L_d del Teorema 6.9.1, de lo cual se deduciría que L_d es Turing-aceptable, contradiciendo así la conclusión de dicho teorema.

Con una entrada $w \in \Sigma^*$, la máquina $\mathcal{M}'$ procede así: enumera sistemáticamente las cadenas $w_1, w_2, w_3, \ldots$ (en el orden lexicográfico, con la subrutina de la sección **??**) hasta que encuentra un k tal que $w = w_k$. Luego simula (o invoca) a $\mathcal{M}$ con entrada $\langle M_k\rangle\langle w_k\rangle$: si $\mathcal{M}$ se detiene en un estado que no es de aceptación, entonces M_k no acepta a w_k pero $\mathcal{M}'$ sí la acepta. Por consiguiente, $L(\mathcal{M}') = L_d$. Esto significa que L_d es Turing-aceptable, lo cual contradice el Teorema 6.9.1. □

Este teorema implica que existen lenguajes que pueden ser aceptados por MT específicas pero en cualquier MT que los acepte habrá cómputos que nunca terminan (obviamente, los cómputos de las cadenas aceptadas siempre terminan). De este hecho extraemos la siguiente importante conclusión: los cómputos interminables, o bucles infinitos, son inherentes a la computación y no se pueden eliminar de la Teoría de la Computación.