Aspectos generales

Llegado a este punto en el que ya se han estudiado tres de las principales máquinas abstractas: el autómata finito, el autómata finito bidireccional y el autómata con pila, resulta conveniente recordar el esquema de la Figura 6.1 ya visto en el Capítulo 1, que mostraba la jerarquía de estas máquinas y las relaciones entre ellas.

El punto de partida en este esquema es el autómata finito (AF), al cual se le incorporaron recursos adicionales para dar lugar a dos nuevos autómatas: el autómata con pila (AP) y el autómata finito bidireccional (AFDB). En el primer caso, se incorporó una memoria LIFO, que tuvo fuerte impacto, ya que le dio al AP la capacidad de reconocer lenguajes generados por gramáticas tipo 2. Esta máquina resultó la base de los analizadores sintácticos, que reconocen a los modernos lenguajes de programación y constituyen una parte central de sus compiladores. El segundo caso pareció menos trascendental ya que al incorporar el movimiento del cabezal en dos sentidos, solo pudo obtenerse alguna mejora en la eficiencia del AF, pero ninguna capacidad adicional. Sin embargo, llegó la hora de comprobar la importancia de esta mejora, al servir el AFDB de base a una nueva máquina, *autómata linealmente acotado*, que a su vez conduce a la *máquina de Turing*.

Así el autómata finito bidireccional es un autómata finito al que se le incorporó la posibilidad de seleccionar el sentido del próximo movimiento del cabezal en cada transición. Como se recordará, fue necesario limitar la cinta de entrada y, para ello, se definió un nuevo alfabeto de cinta Γ, formado por el alfabeto de entrada Σ_E y dos símbolos especiales destinados a demarcar los extremos del medio de lectura ($\vdash$ y $\dashv$). Así, una cadena α a ser procesada es representada en la cinta por $\vdash\alpha\dashv$. Cabe acotar que esos símbolos, reconocidos como BOT (*Begin Of Tape*) y EOT (*End Of Tape*) eran de uso frecuente para demarcar los límites de las cintas magnéticas, de intenso uso en computadores durante mucho tiempo.

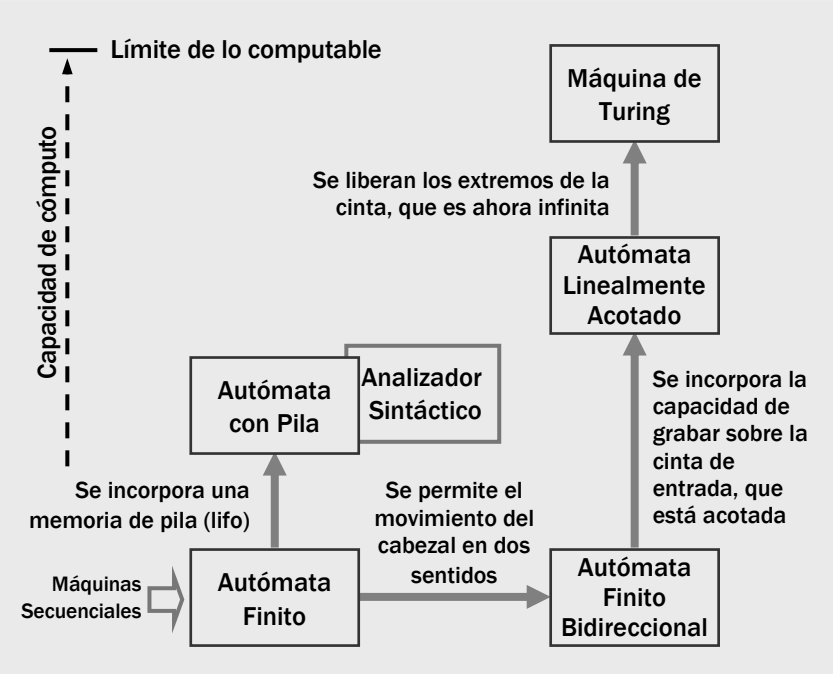

Figura 6.1: Jerarquía de máquinas abstractas.

Como consecuencia de estos nuevos símbolos, la función de transición del AFDB fue ampliada para reconocerlos y también para definir el sentido del movimiento del cabezal en cada posible transición. Para ello, se definió un nuevo alfabeto de movimientos **{I, N, D, P}** que representa las tres posibilidades de desplazamiento del cabezal de entrada: izquierda (I), neutro(N), derecha (D) y la parada de la máquina (P). A partir de todo lo expuesto, el AFDB fue definido como:

$$AFDB = (\Sigma_E, \Gamma, Q, q_0, A, f)$$

donde: $\Gamma = \Sigma_E \cup \{\vdash, \dashv\}$ y donde $f: Q \times \Gamma \rightarrow Q \times \{I, N, D, P\}$[1].

Nótese que la función de transición así definida encierra en realidad dos funciones: una propiamente de transiciones entre estados y otra de movimientos del cabezal. Es decir, ***f*** resume en sí misma a:

$$p=f^T(q,a) \quad y \quad m=f^M(q,a) \quad \text{siendo } p,q \in Q,\ a \in \Gamma,\ m \in \{I,N,D,P\}$$

La novedad es que el autómata finito bidireccional tiene la posibilidad de quedar encerrado en ciclos infinitos, condición que es imposible en los autómatas cuyos cabezales de entrada se mueven regularmente en un mismo sentido. Esta nueva circunstancia no es poca cosa, ya que significa que la operación de este tipo de autómata no necesariamente implica un proceso efectivo o algoritmo.

Autómata linealmente acotado y máquina de Turing

Dando al autómata finito bidireccional la posibilidad de grabar sobre la cinta, se obtiene una nueva máquina abstracta llamada *Autómata Linealmente Acotado* (ALA). Ahora bien, la capacidad de grabar implica la ampliación del alfabeto de cinta Γ con la incorporación de símbolos auxiliares reunidos en un nuevo alfabeto Ω. Como se comprobará en los ejercicios, los símbolos auxiliares agrupados en Ω son una consecuencia natural de la capacidad de grabación que ahora exhibe la nueva máquina abstracta, que le permite al propio autómata incorporar otros símbolos en la salida y distinguir si un cierto símbolo fue o no procesado. En este último caso, el cabezal puede pasar varias veces sobre un mismo símbolo y ser necesario distinguir sucesivas condiciones a lo largo del proceso.

El autómata linealmente acotado queda definido como:

$$ALA = (\Sigma_E, \Gamma, Q, q_0, A, f)$$

donde:

- Σ_E **:** Alfabeto de símbolos de entrada
- Γ **:** Alfabeto de cinta, $\Gamma = \Sigma_E \cup \{\vdash, \dashv\} \cup \Omega$
- Q **:** Conjunto finito, no vacío, de estados posibles.
- q_0 **:** Estado inicial de operación, $q_0 \in Q$
- A **:** Conjunto de estados de aceptación, $A \subseteq Q$
- f **:** Función de transición, $f: Q \times \Gamma \rightarrow Q \times \Gamma \times \{I, N, D, P\}$

Como puede observarse, cuatro de los componentes son los mismos del AFDB: Σ_E, Q, q_0 y A. El alfabeto de cinta Γ fue ampliado con símbolos auxiliares Ω y la otra variante está en la función de transición, que debe incorporar las previsiones para efectuar la grabación de un símbolo sobre la cinta en cada intervalo de tiempo. Nótese que no se trata más de una cinta de entrada sino más bien de una cinta de trabajo o medio de entrada/salida, de largo igual al de la cadena a ser procesada.

Esto significa que la llamada *función de transición* encierra ahora en realidad tres funciones: la de transiciones entre estados (f^T), la de movimientos del cabezal (f^M) y una nueva que es la función de salida (f^S). Las dos primeras ya fueron definidas y la última se expresa como:

$$b = f^S(q, a) \quad \text{con} \quad q \in Q,\ a \in \Gamma \text{ y } b \in \Gamma$$

Ahora bien, si se permite a la cinta extenderse más allá de la cadena a ser procesada, es necesario eliminar las marcas que la limitan. En este caso, se debe definir un símbolo que, por defecto, ocupará el resto del medio de entrada/salida y usualmente se utiliza para este fin el ***b*** (blanco). La ausencia de límites implica que este medio se extiende ahora hasta el infinito,

[1] Respecto de la definición dada en el Capítulo 3, aquí se incorpora una nueva acción que el autómata puede ejecutar en lugar de mover su cabezal, la instrucción **P** de parada. Esto no altera significativamente la esencia de la máquina AFDB, pero es útil a los fines de presentar la definición del ALA y la MT que se está introduciendo.

conformando una enorme área de trabajo o memoria auxiliar. Con esta muy importante variante, el autómata linealmente acotado se convierte en una *Máquina de Turing*:

$$MT = (\Sigma_E, \Gamma, Q, q_0, A, f, \not{b})$$

Los componentes de la MT son los mismos ya definidos para el ALA, con la única excepción del alfabeto de cinta, que debe ser redefinido ya que se eliminaron las marcas de inicio y fin de cinta. Además, debe observarse que en la definición de la máquina de Turing se incluye el símbolo $\not{b}$, que por defecto ocupa el resto de la cinta no utilizada. El alfabeto de cinta de la MT es entonces $\Gamma = \Sigma_E \cup \Omega \cup \{\not{b}\}$.

A lo largo del libro, a través de un proceso que comenzó con la presentación de las máquinas secuenciales y el autómata finito, se llegó al modelo computacional introducido por Alan Turing en su trabajo de 1936 y publicado por la *Sociedad Matemática de Londres*. Turing ideó un modelo matemático abstracto que formalizó el concepto de algoritmo y con el cual demostró que existían problemas que una máquina no podía resolver. Así Turing respondió a la cuestión planteada por David Hilbert sobre si las matemáticas son *decidibles*, es decir, capaces de resolver cualquier problema a partir de un método axiomático general.

Definiciones referidas al ALA y MT

Configuración

El *Autómata Finito Determinista Bidireccional* (AFDB), el *Autómata Linealmente Acotado* (ALA) y la *Máquina de Turing* (MT) constituyen una clase de máquinas abstractas caracterizadas por la capacidad de mover el cabezal en los dos sentidos sobre el medio de entrada. Como se recordará del Capítulo 3, esto llevó a que en el AFDB desaparezca el concepto de *cadena a ser leída*, que era clásico del AF y AP, y se presente la necesidad de replantear la definición de *configuración* o *descripción instantánea*. En efecto, para la completa definición de la condición en la que se encuentra un AFDB en un instante dado **t** se requieren tres componentes, que son: *i*) el estado actual **q**, *ii*) el contenido de la cinta de entrada, representado en este caso por cierta cadena α y *iii*) la posición del cabezal **k** sobre la misma. Luego:

$$K_t = (q, \vdash\alpha\dashv, k) \quad ; \quad \alpha = \text{constante}$$

La configuración o descripción instantánea del ALA y máquina de Turing MT tiene los mismos componentes, con la salvedad de que, al poder grabar sobre la cinta, alteran progresivamente la cadena de entrada en cada transición, convirtiéndola en sucesivas cadenas $\boldsymbol{\beta_t}$. En el caso del autómata linealmente acotado, las cadenas $\boldsymbol{\beta_t}$ mantienen la misma longitud y su configuración es definida como:

$$K_t = (q, \vdash\beta_t\dashv, k) \quad ; \quad |\beta_t| = cte., \ t = 0, 1, 2, \ldots n$$

Por último, en la máquina de Turing, la configuración o descripción instantánea es definida como:

$$K_t = (q, \beta_t, k) \quad ; \quad \beta_t \neq cte., \ t = 0, 1, 2, \ldots n$$

En este caso, $\boldsymbol{\beta_t}$ representa el contenido de la parte de la cinta que es efectivamente utilizada en cada intervalo de tiempo, dejando de lado los extremos seminfinitos ocupados con $\not{b}$, que pueden ser uno o ambos.

En resumen, en el AFDB la cinta es exclusivamente un medio de entrada y, por lo tanto, permanece inalterado, en el ALA, la cinta es un medio de entrada/salida cuyo sector utilizado tiene un largo que permanece inalterado, por estar acotado, y en la MT la cinta es también un medio de entrada/salida, pero que durante su operación cambia en contenido y extensión. Esto último permite disponer de todo el espacio necesario, lo que en la práctica equivale a operar en un medio infinito.

Convenciones de representación

Como ya fue anticipado, tanto en el ALA como en la MT, la función de transición reúne en realidad tres funciones, que son: *i*) transición de estados, *ii*) salida y *iii*) movimiento del cabezal. En la tabla

que representa la función de transición, se deben definir entonces el próximo estado, el símbolo a ser grabado y el movimiento que corresponde a cada condición de operación, separados por comas.

En los grafos, los arcos son etiquetados con la simbología **e/s,m**, que representan el símbolo leído (**e**), el símbolo grabado (**s**) y el código del movimiento del cabezal (**m**). Ejemplos de movimientos de un ALA y de una MT serían los siguientes:

(p, ├0101┤ , 1) ⊢ (q, ├1101┤ , 2) ; (p, 0101, 1) ⊢ (q, 1101, 2)

que implica que la función de transición incluye a ***f*(p, 0) = (q, 1, D)** y que el cabezal pasó de la posición **1** a la posición **2**. Ambos movimientos son representados con el fragmento de grafo que se muestra en la Figura 6.2:

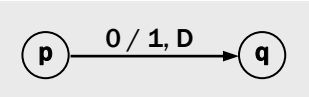

Figura 6.2: Transición del estado **p** al **q** ante la lectura de **0**.

El árbol de configuraciones no suele ser la opción más conveniente para representar el comportamiento de un ALA o una MT al procesar cierta cadena de entrada. En especial, cuando se trata de árboles lineales provenientes del estudio de máquinas abstractas deterministas.

En su lugar, se suele emplear una secuencia de esquemas que muestran el fragmento utilizado de la cinta y la posición del cabezal, que es representado por el símbolo del estado actual. Por ejemplo, si un sector del árbol de configuraciones, mostrado en la Figura 6.3, representa la operación de cierto autómata linealmente acotado a partir de un instante dado, la secuencia de esquemas correspondientes es mostrada en la Figura 6.4.

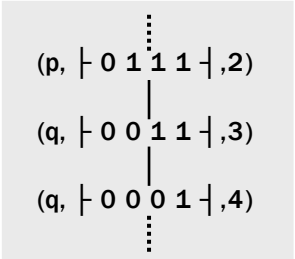

Figura 6.3: Fragmento del árbol de configuraciones de un ALA.

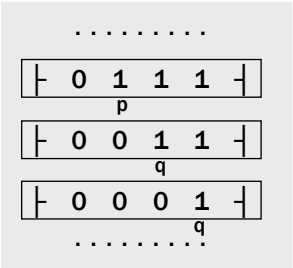

Figura 6.4: Secuencia de esquemas de configuraciones de un ala.

Como pudo apreciarse, se trata de representaciones equivalentes que pueden utilizarse indistintamente, aunque la última está más difundida. Aquí resulta más inmediato visualizar el estado de la máquina en cada instante de tiempo, la posición del cabezal y las sucesivas transformaciones del medio de entrada / salida, representado por la cinta. Cabe aquí reiterar que este recurso es recomendado para autómatas deterministas, ya que no facilita la representación de caminos alternativos o ramificaciones de un árbol.

Interpretaciones del ALA y MT

Los autómatas linealmente acotados y las máquinas de Turing admiten dos interpretaciones: *reconocedoras de lenguajes* y *ejecutoras de procedimientos*.

A. Máquina reconocedora de lenguajes

Un lenguaje **L** definido sobre el alfabeto Σ_E, es reconocido por un ALA o por una máquina de Turing **M** si:

$$L = \{\alpha \,/\, (q_0, \alpha, 1) \vdash^* (q_A, \beta, k),\ q_0 \in Q,\ q_A \in A,\ \alpha \in \Sigma_E^*,\ \beta \in \Gamma^+\}$$

Aquí debe tenerse en cuenta que $\vdash^*$ denota una cantidad finita de movimientos y, por consiguiente, de intervalos de tiempo, lo que implica que ante cualquiera de las cadenas $\alpha \in$ ***L la máquina se detiene*** en un estado de aceptación $q_A \in A$. En este caso, se dice que **L = L(M)**, es decir que el lenguaje definido sobre el alfabeto Σ_E es reconocido por la máquina **M** (en el caso del ALA, **k** debe indicar el fin de cinta ┤).

Por el contrario, si el estado final alcanzado al detenerse no es de aceptación, es decir que $q_A \notin A$, significa que la máquina **M** ha rechazado la cadena α. Puede también darse el caso que la máquina no se detenga con α, entonces se dirá que **M** es indiferente a α ya que ni la acepta ni la rechaza. En esta última condición, la acción de **M** no determina un algoritmo.

B. Máquina ejecutora de procedimientos

Para comenzar, es necesario tener presente que la tarea de ejecutar un procedimiento a partir de cierta cadena α por parte de una máquina abstracta está reservada a autómatas que disponen de la capacidad de grabar sobre la cinta de entrada, como es el caso del ALA y la MT. Es decir, convertir una cadena de entrada en una cadena de salida. En la mayoría de los casos, se tratará de procedimientos efectivos que, a partir de una cantidad finita de movimientos, convertirán la cadena inicial α en otra cadena final **β** que cumplirá ciertas condiciones. Se trata de máquinas que pueden caracterizarse como traductoras, por establecer una relación entre entrada y salida. Para una MT:

$$(q_0, \alpha, 1) \vdash^* (q_f, \beta, k),\ \text{donde}\ q_0, q_f \in Q,\ \alpha \in \Sigma_E^*,\ \beta \in \Gamma^+$$

Sin embargo, hay situaciones en las que el objetivo es el propio proceso y no una cierta cadena final β, tratándose muchas veces de casos en que la máquina cumple infinitas veces un mismo ciclo. En estos casos, interesa la sucesión de estados que recorre reiteradamente el autómata, ya que los mismos suelen estar asociados a condiciones de control. Es decir que la salida del autómata no está representada por la cadena final β sino por la sucesión de los estados alcanzados en un cierto orden. A continuación, se define un proceso que comienza en el estado **p** a partir de cierta cadena inicial **α** y que reiteradamente transita por los estados **p**, **q**, **r**, **s**, **t**. Luego, en el Ejemplo 6.8, se ilustra en detalle uno de estos casos.

$$(p, \alpha, 1) \vdash^\infty \ldots [p, q, r, s, t\,],\quad p, q, r, s, t \in Q,\quad \alpha \in \Sigma_E^*$$

Máquina de Turing modular

El concepto de máquina modular se refiere a la construcción de autómatas mayores a partir de aislar y asignar sus funciones principales a máquinas individuales específicas, que luego son incorporadas a la principal. Esto es, en especial, aplicable a la máquina de Turing ya que se ve facilitado por la disponibilidad de una cinta de E/S infinita.

Esta idea no es otra cosa que una concepción *modular* de una máquina de Turing, concepto análogo al aplicado en el desarrollo e implementación de sistemas convencionales de computación. Este es un enfoque altamente recomendable, como mínimo a nivel conceptual, con el fin de hacer más clara y entendible la operación de la MT a partir de la distinción de sus funciones principales. Es así que en una máquina de Turing modular su función de transición aparecerá sectorizada en correspondencia con los módulos o subprogramas que la componen.

Se brindan, a continuación, algunos ejemplos, comenzando por una *máquina de Turing reconocedora*, en concordancia con el orden en que fueron definidas en el punto anterior las interpretaciones que admite este tipo de máquinas abstractas.

Ejemplo 6.1

Se presenta el ejemplo clásico de reconocer a través de una máquina de Turing el lenguaje dependiente del contexto **L = {$c^n d^n e^n$ / n > 0}**. Para ello, se propone el autómata *MT* = (Σ_E, Γ, Q, q_0, A, *f*, ~~*b*~~), donde:

Σ_E = {c, d, e} Ω = {X, Y, Z} q_0 = p

Γ = {c, d, e} U Ω U {~~*b*~~} Q = {p, q, r, s, t, u} A = {u}

La definición se completaría con la función de transición pero, en este caso, se comienza por plantear su grafo, que es presentado en la Figura 6.5. Como allí puede observarse, la lógica implementada verifica la repetición de secuencias **c..d..e..** y a medida que estos símbolos son encontrados son reemplazados por los símbolos auxiliares **X**, **Y** y **Z**. El autómata sale del ciclo de revisión de la cadena al regresar al estado **p** y encontrar que no hay más símbolos **c**. En ese caso, recorre la cadena hacia la derecha y se detiene en el primer espacio en blanco. Por el contrario, la ausencia de alguno de los tres símbolos en la secuencia impide que la máquina arribe a su estado de aceptación. Se sugiere al lector verificar que esto último es cierto, probando la operación de la MT con diferentes cadenas ajenas al lenguaje.

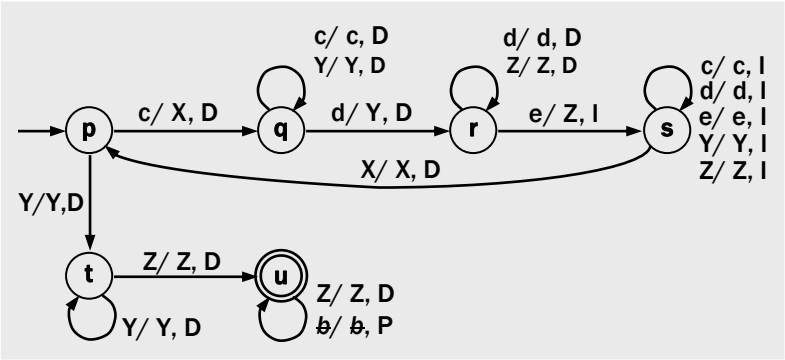

Figura 6.5: Grafo de la MT que acepta cadenas de tipo α = **$c^n d^n e^n$**.

Para completar la definición de esta MT, es necesario presentar su función de transición, como se muestra en la Tabla 6.1:

f	c	d	e	X	Y	Z	~~*b*~~
→p	q,X,D				t,Y,D		
q	q,c,D	r,Y,D			q,Y,D		
r		r,d,D	s,Z,I			r,Z,D	
s	s,c,I	s,d,I	s,e,I	p,X,D	s,Y,I	s,Z,I	
t					t,Y,D	u,Z,D	
*u						u,Z,D	u,~~*b*~~,P

Tabla 6.1: Función de la MT que acepta cadenas de tipo α = **$c^n d^n e^n$**.

La última tarea es comprobar la aceptación de una cierta cadena del lenguaje, por ejemplo: α **= ccddee** y, para ello, se utiliza una secuencia de esquemas de configuraciones, como se muestra en la Figura 6.6.

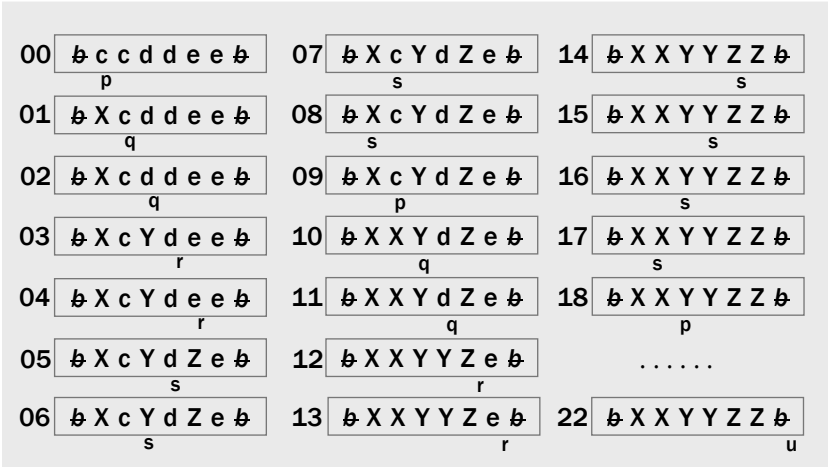

Figura 6.6: Comprobación de la aceptación de α **= ccddee** por la MT.

A partir de un análisis de este primer ejemplo de una máquina de Turing reconocedora de lenguajes, se pueden sacar algunas conclusiones:

a) Para reconocer este lenguaje no es necesario utilizar espacio adicional al ocupado originalmente por las sentencias, lo que significa que para cumplir esta tarea es posible utilizar un autómata linealmente acotado. La definición de este último seria idéntica a la MT presentada, solo que la cadena evaluada debería estar acotada por los símbolos delimitadores $\vdash$ y $\dashv$.

b) Surge naturalmente otra pregunta: ¿pueden este tipo de sentencias ser reconocidas por autómatas de una menor capacidad, por ejemplo autómatas con pila? Para responder con certeza a este interrogante, lo más apropiado es identificar la gramática más simple capaz de generar este lenguaje y determinar su tipo. No sin esfuerzo, se llega a ver que esta gramática tiene las siguientes reglas de producción:

$$P = \{S := cBSe \mid cde,\ Bc := cB,\ Bd := dd\ \}$$

Una vez conocida la gramática, se comprueba con facilidad que no se trata de una gramática independiente de contexto y, por lo tanto, de no haber una más simple que la presentada, su lenguaje no puede ser aceptado por un autómata con pila.

Ejemplo 6.2

Continuando con otro ejemplo de MT reconocedora de lenguajes, se propone desarrollar un autómata que acepte lenguajes de palíndromos, tales como $\mathbf{L = \{\alpha c \alpha^{-1} / \alpha \in \{a, b\}+, \{|\alpha c \alpha\text{-}1| = 2|\alpha|+1\} \cup \{\alpha\alpha^{-1} / \alpha \in \{a, b\}^{+}\}}$, que ya fue objeto del desarrollo de autómatas con pila (Ejemplos 5.1 y 5.2) y analizadores sintácticos (Ejercicio 1) en el Capítulo 5. Algunos de estos AP eran no deterministas, pero si las MT pueden resolver cualquier problema (que tenga solución), seguramente habrá una MT capaz de cumplir la misma función y éste es el objeto del ejemplo que se presenta.

Puede anticiparse que será necesario disponer de símbolos auxiliares para identificar los caracteres ya procesados, agrupados en el alfabeto Ω. También que no será necesario ocupar espacio adicional al de la cadena original, por lo que en este caso se implementara un ALA en lugar de una MT. Cabe recordar que los ALA reconocen lenguajes generados por gramáticas dependientes del contexto, que incluyen a las gramáticas independientes del contexto (que generan los lenguajes reconocidos por los AP), por lo que la elección del ALA debería ser correcta.

Vale la pena comentar que no hay una recomendación definitiva sobre si conviene diseñar un autómata planteando primero su grafo o su función de transición, ya que esto en realidad depende de las preferencias de cada uno. Así como en el Ejemplo 6.1 se propuso un grafo para luego definir la función de transición, aquí se hace lo contrario.

El ALA que se propone como solución, queda definido por los siguientes componentes:

$\Sigma_E = \{a, b, c\}$ $\qquad\qquad$ $\Omega = \{X, Y\}$

Γ = {a, b, c} U Ω U {├, ┤} Q = {p, q, r, s, t, u, v}

q_0 = p A = { v }

La función de transición es representada en la Tabla 6.2 y el grafo del ALA reconocedor es luego presentado en la Figura 6.7.

f	a	b	c	X	Y	├	┤
→p	q,X,D	r,Y,D	p,c,D	v,X,P	v,Y,P		
q	q,a,D	q,b,D	q,c,D	s,X,I	s,Y,I		s,┤,I
r	r,a,D	r,b,D	r,c,D	t,X,I	t,Y,I		t,┤,I
s	u,X,I						
t		u,Y,I					
u	u,a,I	u,b,I	u,c,I	p,X,D	p,Y,D		
*v							

Tabla 6.2: Función de transición del ALA del Ejemplo 6.2.

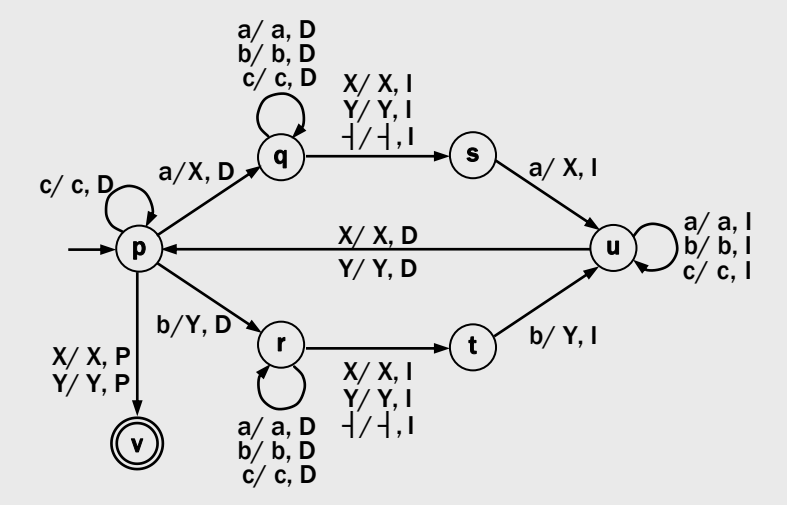

Figura 6.7: Grafo del ALA que reconoce lenguajes de palíndromos.

Es así que un único autómata determinista es capaz de reconocer tanto los palíndromos de largo par como los de largo impar. También vale la pena comentar que pudieron haberse planteado otras lógicas diferentes para resolver este mismo problema. Una de ellas es la de implementar una MT que se comporte como un *AP* y donde la memoria LIFO es almacenada en la cinta, a continuación de la cadena que es evaluada. Se deja al lector definir esta máquina, lo mismo que comprobar el buen funcionamiento de la recientemente diseñada.

Ejemplo 6.3

Se propone ahora diseñar una máquina ejecutora de procedimientos. En este caso, destinada a ordenar en forma creciente los símbolos de las cadenas binarias de cualquier longitud que sean representadas en su cinta. Es decir que una cadena tal como **α = 010101** debe ser convertida en otra cadena **β = 000111** y, para ello, se propone un ALA.

El autómata ordenador queda definido como (Σ_E, Γ, Q, q_0, A, *f*) donde:

Σ_E = {0, 1} Γ = {0, 1, ├, ┤}

Q = {q_0, q_1, s_0, u} q_0 = q_0

A = {u}

En la Tabla 6.3 y en la Figura 6.8, se presentan la función de transición y el grafo del autómata destinado a ordenar cadenas compuestas por ceros y unos.

f	├	0	1	┤
→q_0	q_0, ├,D	q_0,0,D	q_1,1,D	u, ┤,P
q_1		s_0,1, I	q_1,1,D	u, ┤,P
s_0			q_0,0, I	
*u				

Tabla 6.3: Función de transición del ALA ordenador del Ejemplo 6.3.

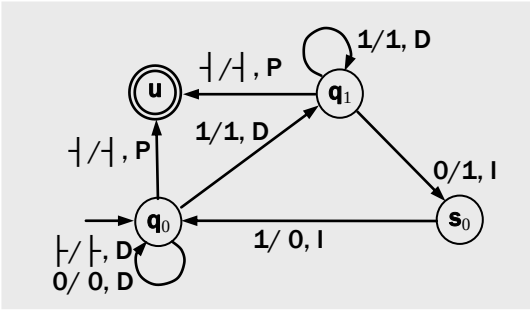

Figura 6.8: Grafo del autómata ordenador.

Como puede observarse, los estados que van siendo adoptados por el autómata representan condiciones específicas de operación, tales como:

q_0 : Lectura de símbolos cero.

q_1 : Lectura de símbolos uno.

s_0 : Después de un símbolo uno se encontró un símbolo cero.

Una vez definido el autómata se evalúa su desempeño. Para ello se muestra a continuación la secuencia de configuraciones del ALA al ordenar la cadena α = 1100 hasta convertirla en la cadena β = 0011, para lo cual demanda 22 intervalos de tiempo. La secuencia de esquemas de configuraciones es representada en la Figura 6.9.

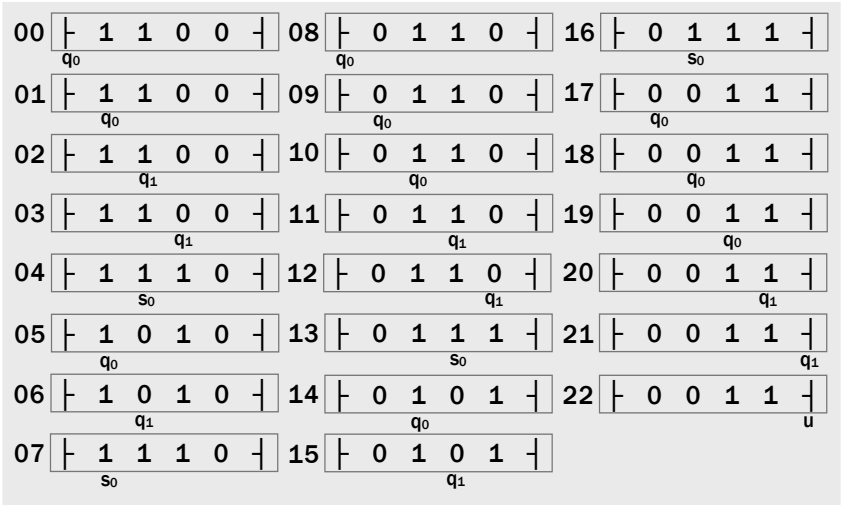

Figura 6.9: Secuencia de esquemas de configuraciones del Ejemplo 6.3.

En este esquema, puede observarse que el comportamiento básico en el proceso de ordenamiento es intercambiar las secuencias **10** por la secuencia **01**. Nótese que, después de cada intercambio, el cabezal vuelve a avanzar buscando la siguiente secuencia **10** o el final de la cadena. Una vez alcanzado el final de la cadena se transita a un estado de aceptación y la máquina se detiene.

Ejemplo 6.4

A continuación, se busca ampliar el autómata del ejemplo anterior para que reordene en forma creciente cadenas con más símbolos. En este caso, definidas sobre el alfabeto **Σ_E = {0, 1, 2}**.

Aquí tampoco es necesario utilizar la cinta más allá del espacio ocupado por la cadena a ser ordenada y, por tal motivo, vuelve a plantearse un autómata linealmente acotado. Es conveniente aclarar que el largo de las cadenas reconocidas no debe estar limitado. Como ejemplo, una cadena tal como **α = 012201** debe ser convertida en la cadena **β = 001122**.

El autómata ordenador queda definido $\boldsymbol{ALA = (\Sigma_E, \Gamma, Q, q_0, A, f)}$, la función de transición es mostrada en la Tabla 6.4 y el grafo es mostrado en la Figura 6.10.

$\Sigma_E = \{0, 1, 2\}$ $\Gamma = \{0, 1, 2, \vdash, \dashv\}$

$Q = \{q_0, q_1, q_2, s_0, s_1, u\}$ $q_0 = q_0$ $A = \{u\}$

f	⊢	0	1	2	⊣
→q_0	q_0, ⊢,D	q_0, 0, D	q_1,1, D	q_2, 2,D	u, ⊣ ,P
q_1		s_0, 1, I	q_1,1, D	q_0, 2,N	u, ⊣ ,P
q_2		s_0, 2, I	s_1, 2, I	q_2, 2,D	u, ⊣ ,P
s_0			q_0, 0, I	q_0, 0, I	
s_1				q_0 ,1, I	
*u					

Tabla 6.4: Función de transición del ALA ordenador.

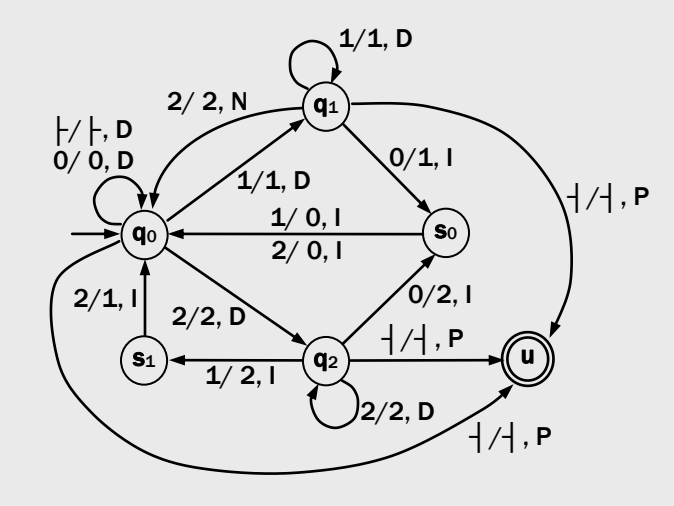

Figura 6.10: Grafo del ALA ordenador.

Como puede observarse, el autómata capaz de reconocer cadenas sobre el alfabeto **{0, 1}** del Ejemplo 6.3 fue ampliado para reconocer ahora cadenas sobre el alfabeto **{0, 1, 2}**. Fue necesario extender la concepción lógica del primer autómata para incorporar la detección de las nuevas secuencias **20** y **21** y convertirlas respectivamente en **02** y **12**. Para ello, fue necesario incorporar dos estados adicionales con el siguiente significado:

q_2 : Lectura de símbolos 2 (dos).

s_1 : Después de un símbolo 2 (dos) se encontró un 1 (uno).

A partir de los nuevos estados, debieron incorporarse las transiciones para reconocer las secuencias ya indicadas y realizar las conversiones necesarias. Aquí se recomienda al lector advertir la presencia de la transición con movimiento *neutro* entre el estado $\mathbf{q_1}$ y $\mathbf{q_0}$, que está etiquetada como **2,2/N**. Nótese que con este recurso se mantiene sin cambios el contenido de la cinta y solo se cambia de estado, de manera de posibilitar la lectura del símbolo **2** desde el estado $\mathbf{q_0}$. Este ejemplo sirve para mostrar que, en programación, los pequeños detalles pueden tener fuerte impacto en las soluciones y para confirmarlo se invita al lector a plantear alternativas que no utilicen movimientos *neutros*.

Una vez que se ha propuesto una solución al problema, en la Figura 6.11, se presenta la secuencia de esquemas de configuraciones que corresponde al procesamiento de la cadena **α = 2012**.

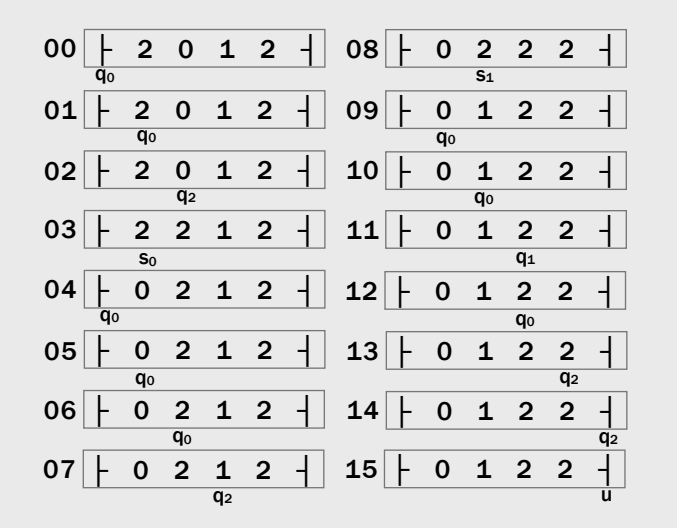

Figura 6.11: Secuencia del esquema de configuraciones del ALA ordenador.

Ejemplo 6.5

Se propone diseñar una máquina ejecutora de procedimientos que duplique las cadenas binarias que estén definidas sobre su cinta. En este caso, la cinta es extendida para alojar la copia, por lo que la tarea estará a cargo de una máquina de Turing. Se prevé un espacio en blanco entre la cadena original y su copia.

La máquina de Turing se define $MT = (\Sigma_E, \Gamma, Q, q_0, A, f, \not b)$, donde

$\Sigma_E = \{0, 1\}$
$\Omega = \{X, Y\}$
$\Gamma = \{0, 1\} \cup \Omega \cup \{\not b\}$
$Q = \{p, q, r, s, t, u, v\}$
$q_0 = p$
$A = \{v\}$

Se comienza por definir el grafo de la máquina de Turing, que es representada en la Figura 6.12. Como puede observarse, se asume que la máquina esta inicialmente en el estado **p** y el cabezal de E/S sobre el primer símbolo de la izquierda. A partir de allí, por cada símbolo binario la MT reproduce el mismo símbolo a la derecha de la cadena original, separada por un espacio en blanco. Esta operación se repite hasta que la totalidad de la cadena original ha sido operada.

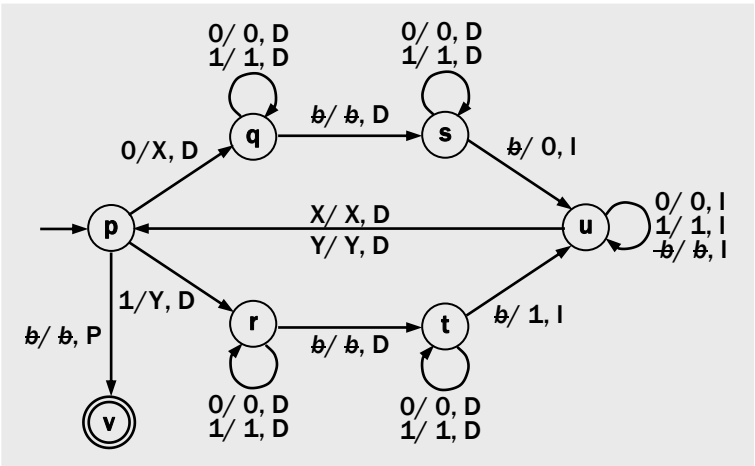

Figura 6.12: Grafo de la MT duplicadora de cadenas binarias.

Para completar la definición formal de la máquina de Turing, hay que describir su función de transición, que es representada en la Tabla 6.5.

f	0	1	*b*	X	Y
→p	q, X,D	r, Y,D	v, *b*,P		
q	q, 0,D	q, 1,D	s, *b*,D		
r	r, 0,D	r, 1,D	t, *b*,D		
s	s, 0,D	s, 1,D	u, 0,I		
t	t, 0,D	t, 1,D	u, 1,I		
u	u, 0,I	u, 1,I	u, *b*,I	p, X,D	p, Y,D
*v					

Tabla 6.5: Función de transición de la MT duplicadora de cadenas.

Como en casos anteriores, se sugiere al lector verificar el buen funcionamiento de esta máquina y también plantear variantes en la lógica de la misma.

Es necesario reconocer que, en los ejemplos presentados, no se han identificado las condiciones de error ni se han definido estados que caractericen estas condiciones. Por tal motivo, las funciones de transición de estas máquinas son *funciones parciales* por no estar definidas para todos los posibles valores de sus argumentos, lo que queda en evidencia por las celdas en blanco en las tablas correspondientes. Es decir que estas MT o ALA llegan a una condición de indefinición ante una cadena a ser rechazada (*un error*), por lo que no se alcanzará el estado de aceptación buscado. Una alternativa es definir explícitamente un estado de error (de no aceptación) y las condiciones que conducen a él, por lo que se sugiere incorporar los estados y condiciones de error en los ejemplos anteriores.

Se presentan, a continuación, dos ejemplos poco convencionales, en los que los autómatas tienen la finalidad de medir intervalos de tiempo operando como cronómetros y temporizadores. Son dos conceptos muy similares, pero distintos. En el primer caso, se desea medir un intervalo de tiempo arbitrario (cronómetro), y en el segundo, reconocer el cumplimiento de un intervalo predefinido (temporizador).

Ejemplo 6.6

En este ejemplo se utiliza un ALA para representar un cronómetro (contador incremental) que suma unidades de tiempo a partir del valor inicial representado por cuatro dígitos sobre su cinta. El cronómetro se detendrá al alcanzarse el máximo valor que se puede representar **0000** (9999 + 1).

El autómata linealmente acotado propuesto queda definido a continuación, con la función de transición mostrada en la Tabla 6.6.

ALA = (Σ_E, Γ, Q, q_0, A, *f*), donde:
Σ_E = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
Γ = Σ_E U { ├, ┤ }
Q = {p, q, r, s}
qo = p
A = {s}

Cabe aquí aclarar que con el fin de reducir la cantidad de columnas de la tabla, se utiliza el símbolo genérico “n” para representar valores en el rango [0..8] y “n+1” para hacer referencia al valor siguiente en cada caso:

f	├	n	9	┤
→ p	p, ├, D	p, n, D	p, 9, D	q, ┤ , I
q		q, n+1, N	r, 0, I	
r	s, ├, P	p, n+1, D	r, 0, I	
*s				

Tabla 6.6: Función de transición del cronómetro del Ejemplo 6.6.

El grafo del cronómetro representado con el ALA es mostrado en la Figura 6.13:

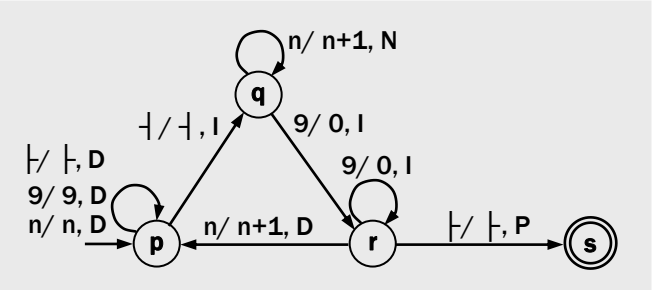

Figura 6.13: Grafo del cronómetro del Ejemplo 6.6.

Como ejemplo, se estudia el comportamiento del ALA cuando se inicia el cronometro con un valor **9997** en la cinta y para representar el proceso se utiliza una secuencia de esquemas de configuraciones, como se muestra en la Figura 6.14. El número que se presenta a la izquierda indica la transición y como ya fue anticipado un mismo símbolo debajo de la cinta indica el estado y posición del cabezal en cada caso.

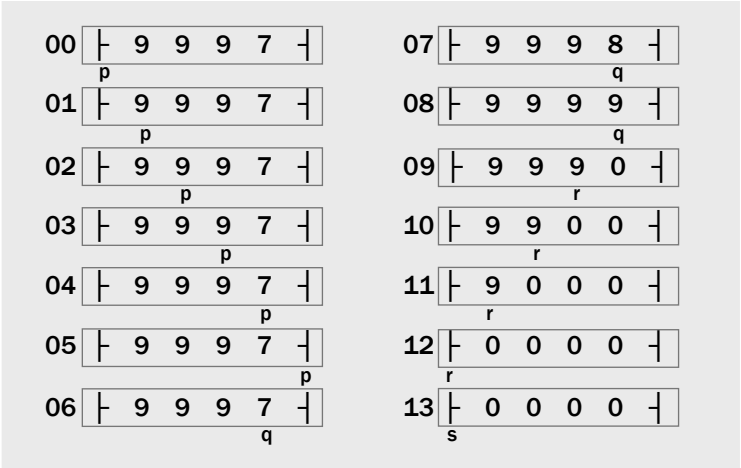

Figura 6.14: Secuencia de configuraciones del cronómetro.

La interpretación de la función de cada estado es la siguiente:

p : posicionar el cabezal en el dígito menos significativo.

q : incrementar el dígito menos significativo hasta alcanzar nueve.

r : grabar cero, retroceder el cabezal e incrementar el dígito en uno.

Ejemplo 6.7

Representar un temporizador (contador decremental) con un ALA que resta unidades de tiempo a partir de un valor inicial predefinido que es representado por cuatro dígitos sobre la cinta. El temporizador se detiene al alcanzar el valor **0000**.

Al igual que en el Ejemplo 6.6, el *ALA* queda definido por los siguientes componentes:

$\Sigma_E = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ $\Gamma = \Sigma_E \cup \{\vdash, \dashv\}$

$Q = \{p, q, r, s\}$ $q_0 = p$ $A = \{s\}$

y la función de transición es mostrada en la Tabla 6.7. Como en el ejemplo anterior, se utiliza un símbolo genérico "**n**" para reducir la cantidad de columnas de la tabla. En este caso, "**n**" representa valores en el rango [1..9] y "**n-1**" hace referencia a un número anterior a "**n**".

f	├	n	0	┤
→ p	p, ├, D	p, n, D	p, 0, D	q, ┤, I
q		q, n-1, N	r, 9, I	
r	s, ├, P	p, n-1, D	r, 9, I	
*s				

Tabla 6.7: Función de transición del contador decremental o temporizador.

Se deja al lector la representación del grafo del temporizador y la comprobación de su buen funcionamiento.

Nótese que tanto en el caso del cronómetro como del temporizador, lo que interesa es la cantidad de intervalos de tiempo (transiciones o movimientos) transcurridos desde que la máquina comienza a operar sobre la cadena **α** hasta que alcanza cierta cadena específica **β**.

Ejemplo 6.8

Al reconocerse que la ejecución de procedimientos es una de las dos principales interpretaciones de la máquina de Turing, se mencionó que hay casos en que no interesa la cadena de salida, sino la sucesión de estados que adopta reiteradamente el autómata, ya que aquí los estados están asociados a condiciones de control. Es decir que la salida del autómata no es una cadena final β sino una sucesión de los estados que son reiteradamente alcanzados en un cierto orden. La mayoría de los electrodomésticos, tales como los hornos de microondas y lavarropas automáticos, son claros ejemplos de casos en los que ciertas acciones de control (activación de campos magnéticos, apertura de válvulas de agua, encendido de un motor, etc.) están asociados al estado de operación del equipo en cuestión. Naturalmente, detrás de ese automatismo, se ha definido una máquina de estados que fue implementada en algún lenguaje de programación y fue cargada en la memoria ROM del microcontrolador del dispositivo. Este es un caso muy simple, y ampliamente difundido, de lo que se denomina *sistema embebido*, importantísimo campo del desarrollo de software.

A efectos de ilustrar este tipo de aplicaciones a través de un ejemplo concreto, se presentará un autómata linealmente acotado que tiene la misión de controlar un semáforo vial. Es decir activar tres luces diferentes (verde, amarillo y rojo) durante ciertos intervalos de tiempo.

Para implementar la secuencia de colores se adoptan los estados $\mathbf{q_V}$, $\mathbf{q_A}$ y $\mathbf{q_R}$, que debe cada uno permanecer activo cierta cantidad de unidades de tiempo, representados por las constantes $\mathbf{T_V}$ (verde), $\mathbf{T_A}$ (amarillo) y $\mathbf{T_R}$ (rojo) respectivamente.

Este autómata opera en forma indefinida, sin detenerse nunca, para lo cual se implementa un ALA con la siguiente definición:

$\Sigma_E = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ $\Gamma = \Sigma_E \cup \{ \vdash, \dashv \}$

$Q = \{q_V, q_A, q_R\}$ $q_o = q_V$

$A = \{ \}$ $T_V = 8, T_A = 1, T_R = 9$

La función de transición es mostrada en la Tabla 6.8 en la que se utiliza el símbolo genérico **n** para representar valores en el rango [1..9].

f	├	0	n	┤
→q_V	q_V, ├, D	q_A , T_A, N	q_V, n-1, N	
q_A		q_R , T_R, N	q_A, n-1, N	
q_R		q_V , T_V, N	q_R, n-1, N	

Tabla 6.8: Función de transición del semáforo vial.

El grafo del ALA de control del semáforo toma la forma:

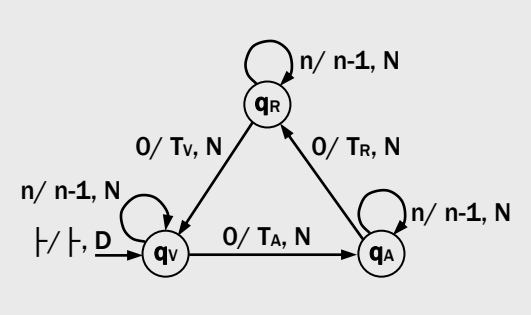

Figura 6.15: Grafo del ALA de control del semáforo del Ejemplo 6.8.

Sobre el problema resuelto, es necesario hacer algunas observaciones:

- Se trata de un ALA que ejecuta un procedimiento de control en el que cada estado activa uno de los colores del semáforo.
- La máquina opera en un ciclo infinito, por lo que no tiene previsto una condición de finalización ni estado de aceptación.
- Se utiliza una única posición de la cinta de E/S con el fin de almacenar el contador y el cabezal permanece inmóvil sobre el mismo.
- El tiempo de permanencia de cada color del semáforo es representado por las constantes $\mathbf{T_V}$, $\mathbf{T_A}$ y $\mathbf{T_R}$ que toman valores específicos según cada caso.

Máquina de Turing No Determinista (MTND)

La máquina de Turing convencional, tal como fue propuesta por su creador en 1936, es determinista. Es decir que cada par (estado, símbolo leído) está relacionado con una única terna (próximo estado, símbolo grabado, movimiento del cabezal) a través de la función de transición.

En caso de que exista para al menos un par (estado, símbolo leído) más de una terna (próximo estado, símbolo grabado, movimiento del cabezal), se está ante una Máquina de Turing No Determinista (MTND). En este caso, la relación de transición es definida:

$$f: Q \times \Gamma \rightarrow P(Q \times \Gamma \times \{I, N, D, P\})$$

Hay aquí dos aspectos que deben ser observados: *i*) dado que la MTD (determinista) convencional es capaz de resolver todo problema que tenga solución, toda MTND podrá ser representada por una MT determinista equivalente, y *ii*) mientras que una MT determinista acepta la entrada siguiendo un único camino computacional, la MTND debe explorar un árbol de opciones computacionales para alcanzar el mismo fin. Sin embargo, el *no determinismo* puede quedar justificado en la necesidad de simplificar y facilitar la interpretación de la máquina. Es decir, un motivo similar al ya expuesto para justificar los AFND en el Capítulo 4.

En resumen, la capacidad de cómputo de la MTD y la MTND será equivalente, en el sentido de que serán capaces de reconocer los mismos lenguajes o hacer las mismas transformaciones. No obstante, para un mismo problema, la velocidad de ejecución podrá no ser la misma.

Si la máquina opera secuencialmente y la MTD reconoce cierta palabra de largo **n** en un tiempo de orden **t(n)**, una MTND requerirá un tiempo máximo de orden $\mathbf{2^{t(n)}}$ (luego se aclarará qué significa *orden*). Esto resulta así, por tener que analizar secuencialmente un *árbol computacional* y la entrada será aceptada si cualquiera de las ramas del árbol finaliza en un estado de aceptación. Para explorar exhaustivamente este árbol ramificado, que se pone de manifiesto al representar las descripciones instantáneas (o configuraciones), se utilizan diferentes estrategias denominadas *búsquedas primero en anchura* y *búsquedas primero en profundidad*, según el orden en que el árbol es explorado.

Por el contrario, si la máquina pudiese operar en paralelo, los resultados se invierten. El no determinismo permitiría reducir el tiempo demandado por la solución de los problemas y resolver problemas de complejidad exponencial en un tiempo polinómico. El concepto de complejidad computacional será tratado más adelante, en el apartado “Complejidad de la máquina de Turing”.

Por último, queda otro aspecto por considerar y es el referido a las transiciones λ (transiciones espontáneas entre estados *sin leer* en la cinta) en las máquinas de Turing. Existiendo en las MT el movimiento *neutro*, es decir, la posibilidad de cambiar de estado sin alterar la posición del cabezal, ¿se justifican las transiciones λ? Se deja al lector analizar las posibles respuestas a este interrogante y sus justificaciones.

Ejemplo 6.9

Se desea reconocer el lenguaje **$L = \{\alpha\alpha^{-1} / \alpha \in \{a,b\}^*\}$** mediante una máquina de Turing que tenga la misión de comprobar que todas las cadenas de un lenguaje son palíndromos, operando desde el centro hacia fuera y desconociendo anticipadamente la posición de los centros de las mismas.

En este caso, una opción es prever una máquina previa que identifique primero el centro de la cadena evaluada, para luego activar una segunda máquina que controle su composición, es decir, verificar que se trate de un palíndromo.

La segunda opción para resolver este problema es una MTND, que no necesita conocer anticipadamente el centro de la cadena evaluada, y es éste el procedimiento adoptado a continuación. Se propone entonces la MTND que tiene la siguiente definición, con la relación de transición mostrada en la Tabla 6.9:

$\Sigma_E = \{a, b\}$ $\Gamma = \Sigma_E \cup \{\not b\} \cup \{A, B\}$

$Q = \{p, q, r, s, t, u, v\}$ $q_o = p$ $A = \{v\}$

f	a	b	A	B	$\not b$
→p	p, a, D q, A, D	p, b, D r, B, D			u, $\not b$, P
q	s, A, I	u, b, P	q, A, D	q, B, D	
r	u, a, P	s, B, I	r, A, D	r, B, D	
s	q, A, D	r, B, D	s, A, I	s, B, I	t, $\not b$, P
t			t, a, D	t, b, D	v, $\not b$, P
u					
*v					

Tabla 6.9: Relación de transición *f* de la MTND reconocedora de palíndromos.

La representación del grafo de la MTND se presenta en la Figura 6.16 y el árbol de descripciones instantáneas que corresponde a la validación de la cadena **α = ababbaba** se muestra en la Figura 6.17:

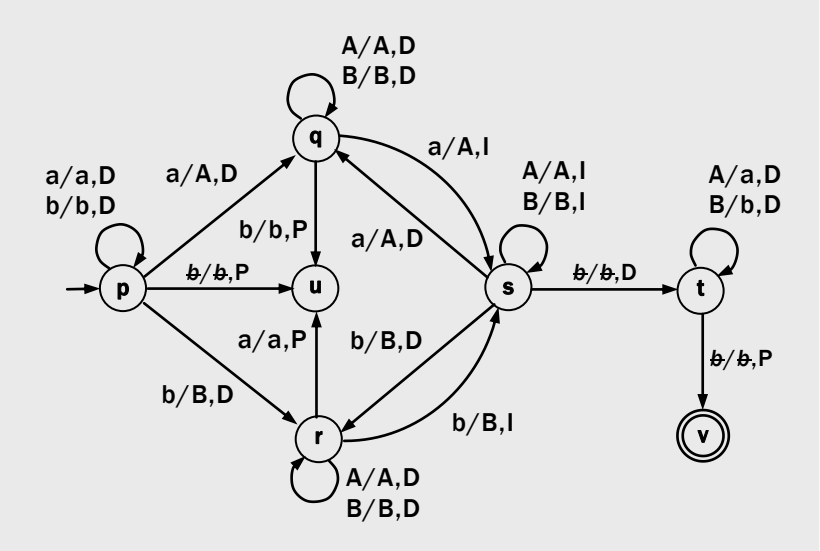

Figura 6.16: Grafo de la MT para aceptar cadenas tipo **α = ababbaba**.

El problema que resuelve esta MTND es la necesidad de comparar entre sí las dos mitades de una

cadena, desconociendo el centro de la misma. Por esta razón, ante cada lectura el autómata presume que es esa la mitad de la cadena y verifica que el siguiente símbolo sea su simétrico.

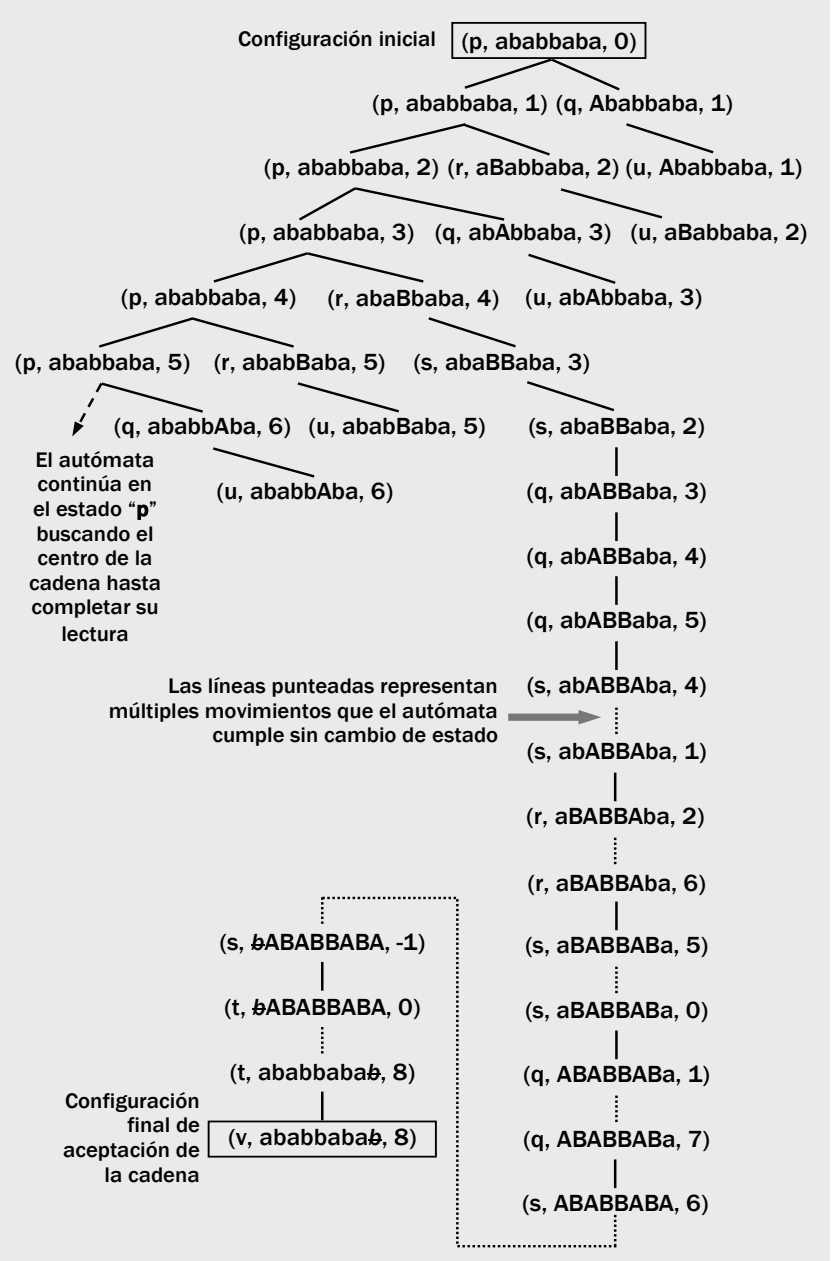

Figura 6.17: Árbol de aceptación de la cadena α **= ababbaba**.

Para el ejemplo presentado, se seleccionó una cadena con solo dos símbolos sucesivos repetidos para reducir las ramificaciones del árbol de descripciones instantáneas, pero es fácil advertir que al aumentar la cantidad de símbolos sucesivos repetidos el árbol crece y se torna cada vez más grande.

Con este ejemplo, se buscó mostrar la definición y operación de una MTND en un problema en el que hay soluciones mucho más eficientes. En efecto, este es el caso de la MTD del ejemplo 6.2, en el que se verifica al palíndromo desde afuera hacia adentro, no siendo entonces necesario conocer su punto medio. Otras alternativas también han sido ya comentadas, como son los casos de emular con una MTD el comportamiento de un AP o emplear una MTD para identificar el punto medio de la cadena y con posterioridad validarla. La enseñanza es que definir máquinas abstractas eficientes, al igual que desarrollar buen software, implica explorar y evaluar alternativas.

Variantes de la máquina de Turing

Combinación de máquinas de Turing

Con anterioridad se trató el concepto de máquina de Turing modular, que da lugar a una función de transición sectorizada que encierra módulos, donde cada uno de ellos tiene un objetivo claramente delimitado.

Una alternativa a este enfoque es plantear máquinas independientes que compartan una misma cinta con el fin de operar en forma sucesiva sobre la misma cadena. De esta manera, cada MT tiene un fin específico y opera a partir de la cadena dejada por la máquina que trabajó con anterioridad, hasta que la última MT en la secuencia alcance su estado de aceptación. Naturalmente, el desdoblamiento de un problema en otros más simples es una estrategia siempre recomendable y presente en ingeniería.

En el Ejemplo 6.2, se utilizó un ALA para validar palíndromos, tantos de largo par como impar. Una alternativa a la lógica planteada en aquella oportunidad, que consistía en trabajar sobre la cadena desde los extremos hacia el centro, es hacerlo en sentido inverso, desde el centro hacia los extremos. Pero aquí, para el caso de palíndromos de largo par, es necesario identificar las dos mitades, y una respuesta ventajosa a este problema que evita la MTND del Ejemplo 6.9 lo brinda la combinación de dos MT. La primera MT cumple la función de distinguir las dos mitades, reemplazando en una de ellas los símbolos de entrada por símbolos auxiliares. Una vez completada esta tarea, la segunda MT verifica que se trate verdaderamente de un palíndromo.

Máquina Universal de Turing

Se denomina así a una MT que recibe en su cinta una descripción codificada de otra máquina de Turing y produce como resultado de su ejecución, el mismo resultado que produciría la MT descripta en la cinta.

Esta máquina recibe el nombre de *universal* por ser capaz de simular el comportamiento de cualquier otra máquina de Turing y fue propuesta por el mismo Alan Turing en 1936. Es así que la MTU funciona como una plataforma para interpretar y ejecutar las MT que sean descriptas en su cinta.

Para implementar una MTU, es necesario definir la forma en que la MT es codificada sobre la cinta, pudiendo adoptarse diversos criterios o *protocolos* de representación. Obviamente, deben definirse la cantidad de estados, los elementos de la función de transición y la cadena que debe ser aceptada o procesada. Para ello, se usan separadores que permitan identificar los diversos campos sobre la cinta.

La MTU encierra un concepto muy potente, pero a la vez muy laborioso y de difícil materialización. Una opción es utilizar simuladores específicos para este caso. Los simuladores serán estudiados en el siguiente capítulo.

Máquina de Turing generalizada

Para generalizar la máquina de Turing, se incorpora más "*hardware*" a su versión original. En particular, se diseña esta nueva máquina con un número arbitrario (pero finito) de cintas de entrada/salida que, a su vez, pueden tener múltiples cabezales de lectura/escritura.

Como es habitual, y sin perjuicio de su generalización, esta máquina de Turing debe quedar especificada por su función de transición ***f***, como ocurre con cualquier MT convencional. En efecto, aquí se deben prever las acciones a ser desarrolladas sobre todas las cintas por todos los cabezales de entrada/salida reconocidos por el autómata, lo que puede llegar a hacerla singularmente compleja. Una consecuencia es la necesidad de adoptar precauciones para evitar inconsistencias impensadas hasta ahora, como sería el caso de conductas contradictorias por parte de distintos cabezales, el resguardo de la integridad de los datos para prevenir que dos o más cabezas intenten escribir distintos símbolos sobre la misma posición o que los movimientos de los cabezales conduzcan a ciclos cerrados infinitos, llamados *abrazos mortales*. Todas precauciones habituales en programación.

Una forma de facilitar la definición de las MTG es imponerles algunas restricciones. La más habitual es prever que haya una sola cinta de entrada en la que se permite *leer*, que haya una sola cinta de salida en la que se permite *escribir*, y que en estas cintas de E/S el cabezal solo pueda moverse a la derecha. El resto de las cintas son consideradas *de trabajo*, en las que se permite la lectura y escritura y los cabezales sobre ellas pueden moverse en ambos sentidos.

Puede demostrarse que estas máquinas *multicinta* y *multicabezal* pueden ser simuladas por la máquina de Turing ordinaria, lo que es obvio ya que se sabe que esta última está en capacidad de resolver cualquier problema que tenga solución. Es decir, la máquina de Turing generalizada puede aportar mayor eficiencia o facilidad operativa, pero de ninguna manera implica la incorporación de capacidad adicional para resolver nuevos problemas.

Ejemplo 6.10

En el Ejemplo 6.5, se estudió una máquina ejecutora de procedimientos destinada a duplicar las cadenas binarias definidas sobre su cinta, y aquí parece apropiado proponer una alternativa para cumplir este mismo trabajo, solo que mediante una MTG de dos cabezales. Para comenzar, y debido a las múltiples variantes posibles, es necesario definir las restricciones de la forma en que la MTG operará. En este caso, se proponen dos cabezales, uno de entrada y el otro de salida, que se encuentran inicialmente sobre el primer símbolo de la izquierda de la cadena a ser duplicada. Naturalmente, el resto de la cinta a ambos lados de la sentencia está ocupada por *blancos*.

En esta máquina, debe definirse el movimiento de ambos cabezales, quedando la función de transición compuesta por cuatro funciones básicas que son las siguientes:

$q_{k+1} = f^1(q_k, e_k)$ transición de estados

$s_k = f^2(q_k, e_k)$ salida de la máquina

$me_k = f^3(q_k, e_k)$ movimiento cabezal entrada

$ms_k = f^4(q_k, e_k)$ movimiento cabezal de salida

que, como se anticipó, definen el próximo estado, la salida, el movimiento del cabezal de entrada y el movimiento del cabezal de salida respectivamente. Al reunir las cuatro funciones en una única tabla de transición, se tiene:

$$f(q_k, e_k) = (q_{k+1}, s_k, me_k, ms_k)$$

La *configuración* o *descripción instantánea*, ya definida y estudiada con anterioridad para la MT convencional, en este caso, debe ser redefinida en concordancia con la arquitectura de la MTG propuesta. Por ello, queda expresada a través de cuatro componentes, que son:

$$K_t = (q, \beta_t, k_E, k_S)$$

donde $\mathbf{k_E}$ y $\mathbf{k_S}$ representan las posiciones de los dos cabezales. Volviendo al problema a resolver, al igual que en el Ejemplo 6.5 se prevé un espacio en blanco entre la cadena original y su copia.

La MTG es definida $MT = (\Sigma_E, \Gamma, Q, q_0, A, f, \not{b})$, donde:

$\Sigma_E = \{0, 1\}$ $\Omega = \varnothing$ (vacío)

$\Gamma = \{0, 1\} \cup \Omega \cup \{\not{b}\} = \{0, 1, \not{b}\}$ $Q = \{p, q, r, s\}$

$q_0 = p$ $A = \{s\}$

En la solución propuesta, se debe comenzar por posicionar ambos cabezales en los lugares convenientes para iniciar el proceso, pero como el cabezal de salida no tiene forma de saber *dónde está* (no lee), debe ser acompañado por el cabezal de entrada hasta su posición inicial, y luego este último regresa al primer símbolo de la izquierda de la cadena a ser duplicada. El grafo del autómata es presentado en la Figura 6.18:

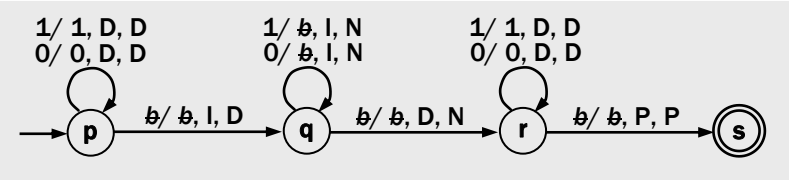

Figura 6.18: Grafo de la MTG duplicadora de cadenas binarias.

Como puede observarse en el grafo de la Figura 6.18, los estados están asociados a las siguientes acciones:

p : mover ambos cabezales, sin alterar la cadena de entrada, hasta encontrar el extremo derecho de la cadena.

q : retroceder solo el cabezal de entrada a su posición inicial, mientras que el cabezal de salida queda sobre el segundo *blanco* a la derecha de la cadena a ser duplicada.

r : avanzar simultáneamente con ambos cabezales hacia la derecha: mientras los símbolos de la cadena son leídos por el cabezal de entrada, son grabados en la cinta por el cabezal de salida.

s :reconocer la detención de la máquina cuando el cabezal de entrada lee *blanco*, que significa que se completó la lectura de la cadena.

Se completa la definición formal de la MTG con la función de transición, que es presentada en la Tabla 6.10:

f	0	1	⊔
→p	p, 0, D, D	p, 1, D, D	q, ⊔, I, D
q	q, ⊔, I, N	q, ⊔, I, N	r, ⊔, D, N
r	r, 0, D, D	r, 1, D, D	p, ⊔, P, P
*s			

Tabla 6.10: Relación de transición de la MTG duplicadora de cadenas.

Para ejemplificar la operación de la máquina propuesta, se presentan en la Figura 6.19 los esquemas de configuraciones correspondientes a la duplicación de la cadena α **= 010**:

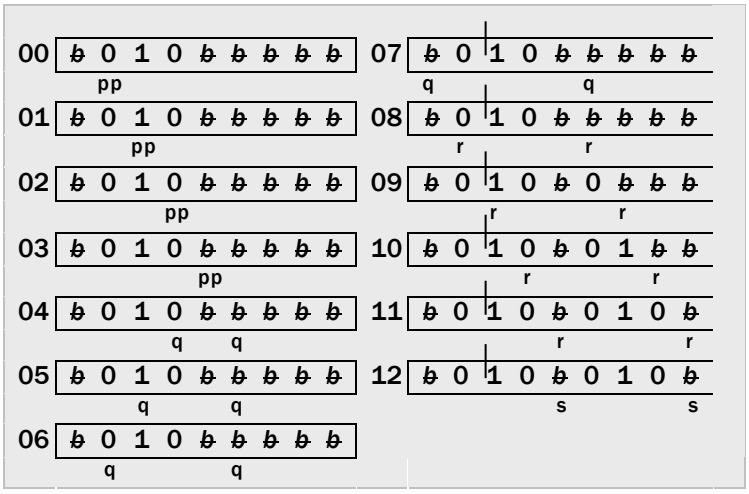

Figura 6.19: Esquemas de configuraciones de la operación de la MTG.

Se recomienda al lector desarrollar el esquema de configuraciones correspondiente a la MT del Ejemplo 6.5 duplicando la cadena α **= 010** y comparar el desempeño de ambas máquinas.

Otras máquinas de Turing generalizadas

En las máquinas de Turing generalizadas presentadas hasta ahora, los medios de entrada/salida son unidimensionales y simbolizados como *cintas*, es decir, medios claramente secuenciales. Una

alternativa para generalizar las MT es incrementar las dimensiones de estos medios, convirtiendo las cintas en planos bidimensionales o aun en espacios tridimensionales. Obviamente, que estos elementos extendidos de E/S deben también estar divididos en casilleros o posiciones de almacenamiento contiguos, solo que distribuidos en dos o tres dimensiones. En estos casos, los movimientos previstos para los cabezales deben ser acordes a las dimensiones de estos medios, resultando que las funciones de transición de la MTG quedan definidas como:

MTG con un cabezal sobre un medio de E/S bidimensional:

$$f: Q \times \Gamma \rightarrow Q \times \Gamma \times \{E, O, N, S, T, P\}$$

MTG con un cabezal sobre un medio de E/S tridimensional:

$$f: Q \times \Gamma \rightarrow Q \times \Gamma \times \{E, O, N, S, U, D, T, P\}$$

donde los movimientos posibles, según el caso, son: (E) este, (O) oeste, (N) norte, (S) sur, (U) arriba, (D) abajo, (T) neutro y (P) parada. Debe observarse que un mayor número de cabezales debería estar acompañado de las previsiones sobre las distintas entradas, sus salidas y los movimientos posibles asignados a cada uno.

Es interesante notar que, al generalizarse los medios de entrada / salida, la *MTG* anticipó conceptualmente los diferentes medios que progresivamente fueron estando disponibles a lo largo del tiempo a medida que se fue desarrollando la tecnología, tales como cintas magnéticas (medios unidimensionales), disquetes (medios bidimen-sionales) y discos rígidos de varios platos y múltiples cabezales (medios tridimensionales). Más recientemente, se hicieron frecuentes las memorias EEPROM y su evolución hacia la memoria *FLASH*, que si bien presentan características técnicas esencialmente distintivas, desde el punto de vista macroscópico de las aplicaciones pueden ser vistas como medios de almacenamiento multidimensionales con diversos modos de direccionamiento, asimilables a los medios conceptuales vinculados a las MTG.

Las máquinas de Turing han sido también generalizadas para incluir la aleatoriedad, disponiendo de transiciones que, en un mismo estado, son elegidas de forma aleatoria y dan lugar a familias de máquinas probabilísticas.

Complejidad de la máquina de Turing

Concepto de complejidad

Desde un punto de vista informático, intuitivamente se asocia el concepto de complejidad con los recursos de cómputo requeridos para resolver un problema, y desde la Ciencia de la Computación, la *Teoría de la Complejidad Computacional* se ocupa de este tema.

La Teoría de la Complejidad Computacional siempre aparece vinculada a la *Teoría de la Computabilidad*, aunque sus objetivos son esencialmente distintos. El foco de la Teoría de la Computabilidad está en la existencia o no de procedimientos que permitan resolver cierto problema, es decir, deja de lado aspectos referidos a la implementación y ejecución de los sistemas computacionales y se concentra en la comprobación de la factibilidad de la existencia de tal solución. En caso afirmativo, se dice que cierta función es computable o calculable. La Teoría de la Complejidad Computacional, por su lado, es la que se ocupa del esfuerzo asociado para completar ese cómputo.

La complejidad computacional fue siempre motivo de interés. En los comienzos de la era de la computación, esto fue una consecuencia natural de los magros recursos disponibles: escasa memoria y reducida capacidad de proceso. Desde entonces y hasta la actualidad, los avances de la tecnología vienen incrementando en forma sostenida la capacidad y calidad de los recursos de computación. Sin embargo, paralelamente, los objetivos de los sistemas informáticos son cada vez más ambiciosos y eso lleva a que el interés por la complejidad computacional mantenga así plena vigencia.

Intuitivamente, se considera que un problema es complejo si su resolución requiere el empleo de un algoritmo complejo. Este último es, a su vez, complejo si su aplicación involucra cálculos complicados y/o su lógica subyacente es complicada. En consecuencia, el concepto de complejidad no parece ser cuantificable o medible, sino más bien subjetivo. Inclusive la complejidad de un mismo problema parece cambiar a medida que el encargado de resolverlo

mejora su capacitación, adopta una actitud apropiada y, más aún, el nivel de complejidad puede variar significativamente dependiendo de la persona que emita el juicio.

Sin embargo, es necesario poder comparar la complejidad de los problemas o confrontar distintas posibles soluciones para resolver uno de ellos, y es éste el campo en el que se desarrolla la Teoría de la Complejidad.

Medidas de la complejidad

Por lo expuesto, es necesario disponer de indicadores de la complejidad intrínseca de los problemas. Indicadores objetivos en los que esté claro el grado de dependencia con la habilidad de la persona a cargo de resolver estos problemas, con los algoritmos que puedan ser utilizados y con las características de los medios de cálculo disponibles. Ésta es la única forma de disponer de indicadores objetivos que brinden valores reconocibles y comparables.

Para asegurar esto último, es decir, la independencia entre las mediciones de los indicadores de complejidad y el medio de cálculo utilizado, se adopta un recurso común para todos los estudios de complejidad y este recurso es la *máquina de Turing*. De esta manera, no solo se dispone de un ambiente bien definido donde trabajar, sino que además las conclusiones obtenidas son fácilmente transferibles a otros sistemas computacionales.

Una de las primeras medidas de complejidad para máquinas de Turing fue la propuesta por Shannon en 1956, que apuntaba a la ***complejidad estructural*** de la máquina y proponía como parámetro al producto del número de estados por el número de símbolos de cinta: **|Q|.|Γ|**. Este enfoque, despertó gran interés en el ámbito de la simulación, donde se destacan los siguientes teoremas:

- ***Teorema 1 de Shannon***: Cualquier máquina de Turing con **m** símbolos y **n** estados puede ser simulada por otra máquina de Turing con exactamente ***dos*** estados y **4*.m.n+m*** símbolos. En particular, existe una máquina universal de Turing de solo dos estados. Además, toda máquina universal de Turing necesita al menos dos estados.
- ***Teorema 2 de Shannon***: Cualquier máquina de Turing con **m** símbolos y **n** estados puede ser simulada por otra con exactamente dos símbolos y menos de **8*.m.n*** estados.

La complejidad estructural propuesta por Shannon no conduce a la determinación de la complejidad operativa de la máquina, por lo que esta propuesta es poco utilizada. Como alternativa, se propuso determinar indirectamente la complejidad de los problemas a partir de la medición de los recursos necesarios para resolverlos. Esta nueva propuesta se apoyó en la obvia convicción que resolver un problema de elevada complejidad demandará más recursos que resolver un problema sencillo. Es así que a partir de entonces se adoptaron el ***tiempo*** y el ***espacio*** como los indicadores más representativos de la complejidad de los problemas.

El primer indicador se refiere a la cantidad de intervalos o unidades elementales de tiempo que demanda completar una computación, es decir, la ejecución de un proceso. Este indicador es definido como ***Complejidad Temporal T*(*n*)** del problema y es expresado en función del tamaño ***n*** del problema o de sus datos.

El segundo indicador es la cantidad de espacio de almacenamiento requerido para resolverlo y se apoya en la idea de que al aumentar la complejidad de un proceso aumenta también el espacio que su solución demanda. Este parámetro es reconocido como la ***Complejidad Espacial, E*(*n*)** del problema y también es expresado en función de ***n***.

Obsérvese que las complejidades temporales y espaciales son parámetros diferentes y normalmente sus valores van a diferir para un mismo cálculo. Sin embargo, no son totalmente independientes ya que en **m** intervalos de tiempo una *MT* tiene acceso a un máximo de **m+1** celdas de su cinta. Es decir que si ***T*(*n*) = *m***, siempre ***E*(*n*) ≤ *m*+1**.

Llegado a este punto, y para un cierto problema dado, es necesario encontrar expresiones que relacionen las complejidades temporales y espaciales de las soluciones con el tamaño de los datos **n**. Se busca algo muy importante, que es conocer la forma en la que crecen las demandas de recursos a medida que crece la dimensión del problema a ser resuelto. Estas expresiones suelen responder a

tres formas generales: la polinomial, la exponencial y la logarítmica, sirviendo de referencia los siguientes ejemplos:

Expresión polinomial: $An^4 + Bn^3 + Cn^2 + Dn + E = O(n^4)$

Expresión exponencial: $A2^n + B = O(2^n)$

Expresión logarítmica: $A \log n + B = O(\log n)$

Como consecuencia de estas expresiones se define la complejidad temporal lineal, polinómica, logarítmica, exponencial, etcétera, según la naturaleza de la relación que la vincula con su dimensión **n**. Por ejemplo, si las unidades de tiempo que demanda la solución de un problema de dimensión **n** es $\boldsymbol{T(n) = 3.n^3 + 6.n + 12}$, se dirá que su complejidad temporal es polinómica, en este caso, de grado tres.

Nótese que estas expresiones tienen un término dominante que se acentúa al crecer la dimensión **n**, hasta tener una importancia excluyente. Se llega así al concepto del ***orden*** de la complejidad de los problemas, que se caracteriza con la letra **O** (denominada *notación O grande*), y se representan por ejemplo como $O(n^3)$, $O(2^n)$ u $O(\log n)$.

Clases de problemas

A partir de las formas generales de las expresiones de complejidad, se clasifican los problemas en las siguientes dos grandes clases:

- ***Clase P***: Incluye los problemas que pueden ser resueltos en tiempo polinómico por una MT determinista. Los problemas de complejidad temporal polinómica son denominados ***tratables***, es decir, pueden ser resueltos en la práctica en un tiempo razonable.
- ***Clase NP***: Incluye los problemas que pueden ser resueltos por una *MT* no determinista en tiempo polinómico, lo que en la práctica conduce en general a una complejidad exponencial. Al llevarse la dimensión del problema **n** a valores de utilidad práctica el tiempo de proceso alcanza valores inimaginables. Estas soluciones carecen por lo tanto de interés y estos problemas son calificados como ***intratables***. Su nombre *NP* proviene del inglés ***N****on-deterministic* ***P****olynomial time*.

Como una MTD es un caso particular de una MTND, claramente se cumple que $P \subseteq NP$. Aún los investigadores no han podido determinar si la inclusión es estricta ($¿P \neq NP?$) y éste es actualmente uno de los problemas no resueltos más importantes de la matemática; para quien lo resuelva, el *Clay Mathematics Institute* ha establecido un premio de un millón de dólares (*The Millenium Prize*). En relación a este problema, se definió (y se investiga con intensidad) la clase de problemas ***NP-completos***, un subconjunto problemas difíciles de NP, a los que todos los otros problemas NP pueden reducirse en tiempo polinómico.

La distinción entre clases no se manifiesta en problemas de pequeño tamaño **n**, donde todos los algoritmos de solución de los mismos parecen ser similares, pero sí se pone de manifiesto cuando los algoritmos son aplicados a datos de gran tamaño, que son los que revisten interés práctico. Es un error grave y frecuente confiar en que la creciente capacidad de las máquinas es suficiente para resolver estos problemas.

Medición de la complejidad

Una vez definidos los indicadores de complejidad, estos deben ser evaluados para cada caso particular. Necesariamente, habrá un problema a resolver y un algoritmo propuesto para alcanzar ese fin, para lo que es necesario reconocer los tres tipos de situaciones que se presentan:

1. La determinación de los indicadores de complejidad temporal y espacial, pueden obtenerse a través de un proceso inductivo que se apoya en un estudio detallado del algoritmo utilizado. Este estudio permitirá definir expresiones generales que serán válidas para cualquier tamaño del problema estudiado, como ocurre en el caso del Ejemplo 6.11 que se verá a continuación.
2. El algoritmo es lo suficientemente enmarañado como para que no sea fácil o posible obtener las expresiones de complejidad a través de su análisis. En estos casos, se hacen

evaluaciones de los indicadores de complejidad resultantes para ciertas dimensiones del problema, se proponen grados para el polinomio que representa su complejidad y se calculan algebraicamente los coeficientes del polinomio. En el Ejemplo 6.12, se determinan las expresiones de complejidad de esta manera. Pueden utilizarse programas específicos (simuladores) para conocer los valores de los indicadores de complejidad, que permitan el posterior cálculo de los coeficientes del polinomio.

3. El caso es similar al anterior, solo que los indicadores de complejidad no dependen únicamente de la dimensión del problema sino también de la forma en la que se presentan los datos. Puede citarse como ejemplo el caso de un algoritmo de ordenamiento de un vector, que reconoce dos condiciones extremas: *i*) el vector ya está previamente ordenado y *ii*) está completamente desordenado. En el caso de la MT del Ejemplo 6.4, las cadenas extremas de entrada estarían representadas por **001122** y **221100**, cuyo ordenamiento obviamente no requiere el mismo esfuerzo. En estos casos, se suelen utilizar dos valores de complejidad: a) el que corresponde a la condición extrema que demanda el máximo esfuerzo y representa una cota superior a la complejidad del problema y b) el de la condición de trabajo más probable (una situación media).

En todos los ejemplos citados, las expresiones de complejidad buscadas tienen las finalidades de permitir: i) conocer los órdenes de las complejidades de los problemas, ii) cuantificar esas complejidades para ciertas dimensiones específicas y poder hacer previsiones referidas a los recursos necesarios para resolverlos y iii) comparar entre sí diferentes problemas o distintas formas de abordar un mismo problema.

En el Ejemplo 6.13, se compara la complejidad temporal de una máquina de Turing destinada a verificar cadenas que están repetidas con un separador (Ejemplo 6.11) con la de otra máquina de Turing que verifica que la segunda cadena sea la inversa de la primera (Ejemplo 6.12).

Para cerrar esta breve introducción a la complejidad computacional resulta justo mencionar a Edsger Dijkstra (1930-2002), quien hizo numerosos y muy importantes aportes a la ciencia de la computación y, en particular, al análisis de la complejidad de algoritmos.

Ejemplo 6.11

Se considera una máquina de Turing destinada a reconocer el lenguaje **$L = \{\alpha\#\alpha \ / \ \alpha \in \{0, 1\}^+\}$**. Para cumplir esta función, la MT propuesta avanza sobre la cinta a partir de cada símbolo del prefijo α hasta superar el separador #, y luego busca confirmar la presencia del mismo símbolo en la misma posición en el sufijo α. Se requiere conocer la complejidad temporal y espacial de este problema.

Se toma como referencia una cadena **$\beta = \alpha\#\alpha$**, donde **$|\alpha| = 4$**, es decir que **$|\beta| = 2|\alpha|+1$.** A partir del largo de la cadena, se deben inducir las expresiones de las complejidades temporal y espacial correspondientes. Para ello, se muestra en la Figura 6.20 un esquema en el que se representan las traslaciones del cabezal y la cantidad de unidades de tiempo que corresponde a cada una en función del largo ***n* = |β| = 9**.

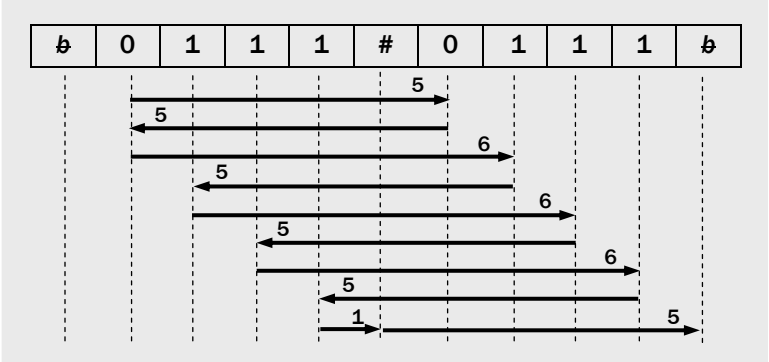

Figura 6.20: Esquema del proceso de aceptación de $\alpha\#\alpha$.

Se analizan los movimientos para expresarlos en función de ***n***, deduciendo:

- Un primer movimiento, desde el primer símbolo del prefijo hasta el primer símbolo del sufijo, que demanda 0,5(*n*+1) unidades de tiempo.
- Los retrocesos desde el sufijo hasta el prefijo. Habrá un total de $|\alpha|$ = 0,5(*n*-1) retrocesos, cada uno de los cuales demanda 0,5(*n*+1) unidades de tiempo.
- Los movimientos de avance de la primera a la segunda cadena. Hay $|\alpha|$ = 0,5(*n*-1) avances con 0,5(*n*+1)+1 unidades de tiempo.

A partir de este análisis, se deduce que la complejidad temporal es:

$$T(n) = \frac{n+1}{2} + \left(\frac{n-1}{2}\right)\left(\frac{n+1}{2}\right) + \left(\frac{n-1}{2}\right)\left[\left(\frac{n+1}{2}\right)+1\right]$$

$$T(n) = \frac{n+1}{2} + \frac{n^2-1}{4} + \frac{n^2-1}{4} + \frac{n-1}{2} = \frac{n^2}{2} + n - \frac{1}{2} = O(n^2)$$

y la complejidad espacial es:

$$E(n) = n = O(n)$$

Para el caso específico en que *n* = 9 se tiene *T*(*n*) = 49 y *E*(*n*) = 9.

Ejemplo 6.12

Se considera ahora la máquina de Turing que reconoce un lenguaje de palíndromos **L = {$\alpha\#\alpha^{-1}$ / $\alpha \in \{0, 1\}^+$},** es decir que el sufijo es la cadena inversa del prefijo. Para resolver este problema, la MT mueve el cabezal hasta el separador #, y a partir de allí avanza desde el medio de la cadena hacia los extremos, verificando los símbolos del sufijo y prefijo con sucesivos movimientos de avance y retroceso. Se requiere conocer la complejidad temporal y espacial de este problema.

En la Figura 6.21, se muestra un esquema en el que se representan las traslaciones del cabezal y la cantidad de unidades de tiempo que corresponde a cada una para el caso en que *n* = 2$|\alpha|$+1 = $|\beta|$ = 9.

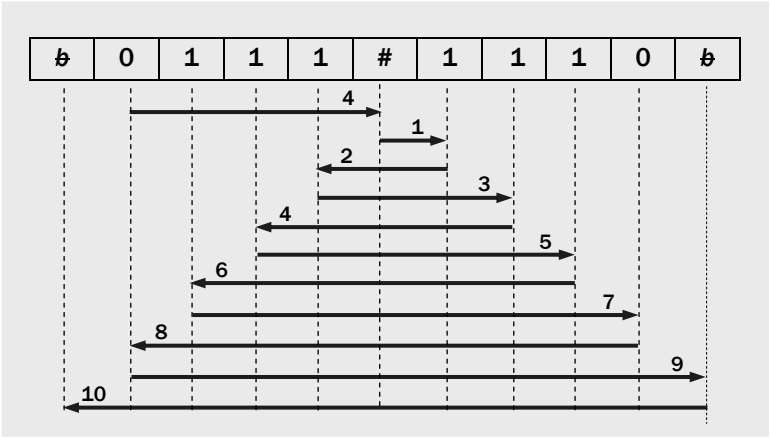

Figura 6.21: Esquema del proceso de aceptación de $\alpha\#\alpha^{-1}$.

A diferencia del caso anterior, los movimientos son aquí progresivamente más largos y la determinación de las expresiones buscadas de complejidad temporal y espacial de una cadena de largo genérico *n* no es tan intuitiva. Tampoco es algo difícil, y se invita al lector a comprobar anticipadamente, siguiendo un procedimiento similar al anterior, que la complejidad temporal del problema es:

$$T(n) = 0{,}5\, n^2 + 2n + 0{,}5 = O(n^2)$$

Sin embargo, hay casos de complejidad verdaderamente elevada y para superar esta dificultad, se recurre al procedimiento ya anticipado como Caso 2 al tratar la "medición de la complejidad" en el apartado anterior. Para ello:

- Se determina el total de intervalos de tiempo necesarios para verificar cadenas de ciertos largos (ejemplo *n* = 3, 5, 7 y 9), incluyendo naturalmente los movimientos de avance y de

retroceso del cabezal. En el análisis, se tuvieron en cuenta los movimientos finales para asegurar que se alcanzaron los ***b*** en ambos extremos. Estos valores son presentados en la Tabla 6.11.

Largo de la cadena	n	3	5	7	9
Intervalos de tiempo	*T(n)*	11	23	39	59

Tabla 6.11: Total de movimientos de avance y retroceso.

- Se calculan los coeficientes del polinomio que responden a estos valores tabulados, asumiendo que se trata de expresiones de tercer grado. Para ello, se recurre al álgebra matricial y se plantea:

$$T(n) = An^3 + Bn^2 + Cn + D$$

$$\begin{bmatrix} 3^3 & 3^2 & 3 & 1 \\ 5^3 & 5^2 & 5 & 1 \\ 7^3 & 7^2 & 7 & 1 \\ 9^3 & 9^2 & 9 & 1 \end{bmatrix} \begin{Bmatrix} A \\ B \\ C \\ D \end{Bmatrix} = \begin{Bmatrix} 11 \\ 23 \\ 39 \\ 59 \end{Bmatrix}$$

La solución del sistema de ecuaciones es la siguiente:

$$\begin{Bmatrix} A \\ B \\ C \\ D \end{Bmatrix} = \begin{Bmatrix} 0{,}00 \\ 0{,}50 \\ 2{,}00 \\ 0{,}50 \end{Bmatrix}$$

El valor nulo del coeficiente *A* significa que la complejidad temporal es correctamente representada por un polinomio de segundo grado, es decir, O(n^2). A partir de los valores de los restantes coeficientes *B*, *C* y *D*, se dispone de la expresión de la complejidad temporal que corresponde a la validación de la cadena:

$$T(n) = \frac{1}{2}n^2 + 2n + \frac{1}{2} = O(n^2)$$

Una segunda alternativa para obtener estas mismas expresiones consiste en recurrir al Excel, obteniendo con este recurso gráfico las curvas y expresiones presentadas en la Figura 6.22.

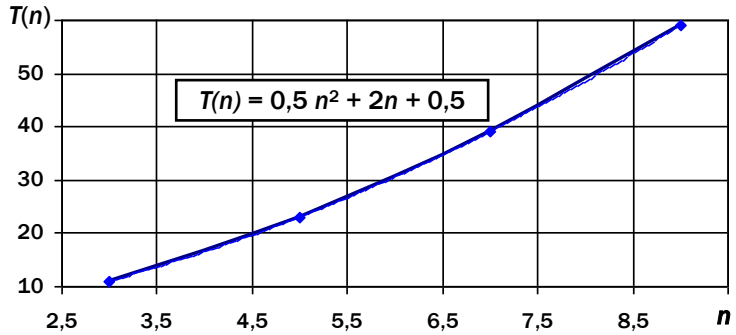

Figura 6.22: Curva de tendencia de T(n).

Para cumplir esta tarea los pasos son: *i*) representar gráficamente en Excel los datos de la Tabla 6.11, *ii*) aproximar las curvas con polinomios de diversos grados hasta obtener representaciones no oscilantes y *iii*) solicitar las expresiones de las fórmulas de las curvas de tendencia.

Por su parte, la máquina no utiliza memoria auxiliar por lo que su complejidad espacial es:

$$E(n) = n = O(n)$$

Nótese que las expresiones obtenidas deben siempre ser utilizadas para valores de ***n*** que estén próximos al intervalo de los casos conocidos, donde aquí fueron **3 ≤ *n* ≤ 9**. Nunca es conveniente hacer predicciones para un valor muy lejano del intervalo simulado (por ejemplo $n > 100$) porque podría ocurrir que en la zona conocida haya varias funciones que representen correctamente los puntos, pero que luego divergen cuando ***n*** crece. En ese caso, la predicción puede arrojar resultados erróneos. La importancia práctica de poder predecir el desempeño de un algoritmo ante volúmenes muy grandes de datos es obvia y no merece mayores comentarios.

Ejemplo 6.13

Comparar las complejidades temporales de las máquinas de Turing que validan las cadenas de las formas generales $\beta = \alpha\#\alpha^{-1}$ y $\beta = \alpha\#\alpha$.

La complejidad temporal obtenida para $\beta=\alpha\#\alpha^{-1}$ es (Ejemplo 6.12):

$$T(n) = \frac{1}{2}n^2 + 2n + \frac{1}{2}$$

y la complejidad temporal obtenida para $\beta = \alpha\#\alpha$ es (Ejemplo 6.11):

$$T(n) = \frac{1}{2}n^2 + n - \frac{1}{2}$$

La relación entre ambas es:

$$R_T(n) = \frac{\frac{n^2}{2} + 2n + \frac{1}{2}}{\frac{n^2}{2} + n - \frac{1}{2}}$$

y considerando problemas de dimensión elevada, lo que en este caso significa que las cadenas analizadas son de gran longitud, se tiene:

$$\lim_{n\to\infty} R_T(n) = \lim_{n\to\infty} \frac{\frac{n^2}{2} + 2n + \frac{1}{2}}{\frac{n^2}{2} + n - \frac{1}{2}} = \lim_{n\to\infty} \frac{\frac{1}{2} + \frac{2}{n} + \frac{1}{2n^2}}{\frac{1}{2} + \frac{1}{n} - \frac{1}{2n^2}} = 1$$

Lo que comprueba que cuando la dimensión de los datos (largo de la cadena estudiada) es grande, las complejidades temporales de ambos problemas son las mismas. Es decir que la relación R_T tiene una tendencia asintótica a uno. Esto se visualiza con claridad en la representación gráfica de R_T en función de ***n*** de la Figura 6.23.

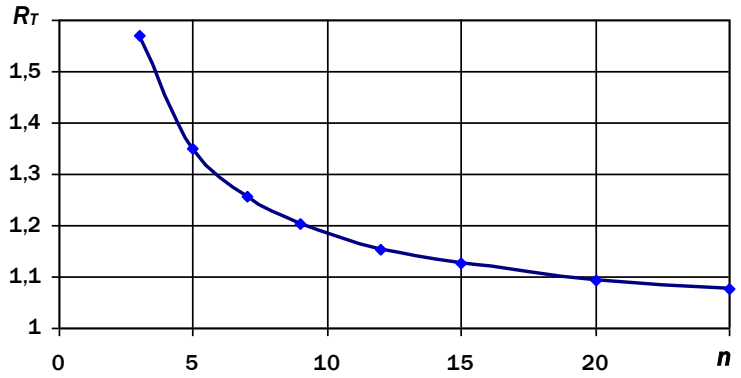

Figura 6.23: Curva de la relación de complejidades de ambos casos.

En resumen, se comprobó que las complejidades operativas de las máquinas de Turing propuestas para resolver ambos problemas son las mismas. Estas complejidades operativas quedaron representadas por la complejidad temporal **T(*n*)** y la complejidad espacial **E(*n*)**.

La conclusión desde el punto de vista del problema tratado es que, verificar la presencia de dos cadenas iguales en la cinta de entrada, ya sea que estén repetidas o una sea la inversa de la otra, demandan el mismo esfuerzo en términos del tiempo de proceso y espacio requerido.

Sería interesante hacer esta misma comparación en base a la complejidad estructural de Shannon, lo que requiere disponer de las tablas de las funciones de transición de ambas máquinas de Turing. Se sugiere al lector hacer esta comparación y sacar sus conclusiones. También cabe reiterar que este mismo tipo de análisis es utilizado cuando se debe identificar el mejor algoritmo para resolver un mismo problema.

Isomorfismos de la máquina de Turing y el ALA con sus gramáticas

Ya fue establecido en el Capítulo 1 que los lenguajes son el vínculo entre las máquinas abstractas y las gramáticas, quedando definidos isomorfismos entre ellos. En el caso de la *máquina de Turing*, los lenguajes reconocidos son denominados *recursivamente enumerables* y provienen de *gramáticas sin restricciones*. Los lenguajes recursivos enumerables son también denominados *estructurados por frases* y constituyen el conjunto de todos los lenguajes que describen procedimientos computacionales. Cabe recordar que fue el propio Chomsky el que estableció en 1959 la equivalencia entre las *gramáticas sin restricciones* o Tipo 0 y las máquinas de Turing. Al presentarse, en el Capítulo 2, la jerarquía de Chomsky se estableció que estas gramáticas no presentaban restricciones en sus reglas de producción, de allí su nombre, salvo la obvia necesidad de que la parte izquierda de la regla debe incluir al menos un símbolo no Terminal. Otro aspecto destacado es que estas gramáticas pueden tener reglas compresoras, incluyendo las no generativas.

Bajando un nivel en la jerarquía de máquinas se encuentra el autómata linealmente acotado y aquí el isomorfismo es planteado con las gramáticas dependientes de contexto, siendo el vínculo los lenguajes generados que llevan su mismo nombre. Aquí las reglas de producción se ven restringidas por la necesidad de un contexto y no se admiten reglas compresoras, salvo que se trate de la regla lambda. Fue Sige-Yuki Kuroda el que en 1964 estableció el vínculo entre los ALA y las gramáticas Tipo 1. Posteriormente propuso una forma normal para estas gramáticas, de la misma manera que Chomsky y Greibach propusieron las formas normales para las gramáticas Tipo 2 que llevan sus nombres. Se denomina Forma Normal de Kuroda.

Al igual que con los autómatas finitos y los autómatas con pila, se han establecido numerosos procedimientos para definir máquinas de Turing y autómatas linealmente acotados a partir de sus gramáticas, e igualmente en sentido inverso se dispone de procedimientos para definir las gramáticas capaces de generar lenguajes a ser reconocidos por ciertas máquinas de Turing o ALA. Sin embargo, entrar en mayor profundidad en estos isomorfismos queda fuera de los objetivos de este libro y al lector interesado en este tema se le sugiere remitirse a la obra de Alfonseca Cubero y otros, referencia 4.

//