Conceptos generales

Tal como fue anticipado en el primer capítulo, el británico Alan M. Turing presenta, en 1936, el modelo de máquina abstracta que lleva su nombre. Por su lado, el ingeniero y matemático estadounidense Claude Shannon comenzó a aplicar, en 1938, la lógica matemática en el análisis de circuitos eléctricos combinatorios y secuenciales, impulsando también el desarrollo de una teoría formalizada sobre las máquinas abstractas. Las máquinas secuenciales más difundidas son las propuestas por George Mealy en 1955 y por Edward Moore en 1956, representando en ambos casos modelos matemáticos sincrónicos de máquinas codificadoras (cifradoras) rudimentarias. Se trata de máquinas secuenciales o máquinas de estados esencialmente traductoras, es decir que a partir de una sucesión de símbolos de entrada generan una sucesión de símbolos de salida. Además, se acepta que estas máquinas estén operando en forma permanente y, por lo tanto, su definición no incluye la identificación de un estado de arranque ni de un estado de detención. Sin embargo, a la hora de determinar el comportamiento de una de estas máquinas ante cierta secuencia de símbolos de entrada, será necesario identificar el estado en el que la máquina se encuentra, en la práctica, un estado inicial.

La formalización de las máquinas secuenciales puede ser ampliada, reconociéndose la existencia de un único estado a partir del cual la máquina entra en servicio (estado inicial) y uno o más estados en los que completa su operación, formando todos ellos parte de un conjunto finito de estados posibles. Y así fue definido el *autómata finito*. En su forma más general, el autómata finito es también traductor, generará una cadena de salida, y al completar la lectura de la cadena de entrada arribará a algún estado final. Es por este motivo que en su operación el autómata finito determina un procedimiento efectivo o algoritmo.

Con el tiempo, el interés se orientó hacia el estudio de la potencialidad de los autómatas finitos como máquinas reconocedoras de lenguajes, con lo que se abandonó la idea de estados finales (de detención) y adquirió preponderancia la existencia de uno o más estados de aceptación, al tiempo que su capacidad traductora pasó a un segundo plano y se justificó la definición de máquinas sin capacidad de salida.

El siguiente paso fue dotar al autómata finito de la capacidad de mover su cabezal de lectura en ambos sentidos, lo que implica poder releer la cadena de entrada. Esto no incorpora ninguna capacidad computacional adicional al autómata, pero sí la posibilidad de un modelado más sencillo. También un riesgo, ya que la selección del sentido del movimiento del cabezal posibilita que el autómata quede encerrado en un ciclo infinito. Estos autómatas finitos, llamados *bidireccionales*, no tienen una difusión muy grande, pero establecen conceptualmente un camino que conduce a las máquinas de Turing.

Máquinas secuenciales

Máquina de Mealy

Para las máquinas abstractas o máquinas de estados más simples, se reserva la denominación de *máquinas secuenciales*, reconociéndose dos variantes principales que son la máquina de Mealy y la máquina de Moore. La *máquina de Mealy* tiene cinco componentes y es definida así:

$$ME = (\Sigma_E, \Sigma_S, Q, f, g)$$

donde:

- Σ_E **:** Alfabeto de símbolos de entrada
- Σ_S **:** Alfabeto de símbolos de salida
- Q **:** Conjunto finito y no vacío, de estados posibles.
- f **:** Función de transición, $f: Q \times \Sigma_E \rightarrow Q$
- g **:** Función de salida, $g: Q \times \Sigma_E \rightarrow \Sigma_S$

La función de transición ***f*** tiene la finalidad de definir el próximo estado que adoptará la máquina a partir de su estado actual y cada uno de los posibles símbolos de entrada. De igual forma, la función de salida ***g*** define la salida de la máquina a partir de los mismos argumentos.

Por tratarse de funciones de dos argumentos, resulta conveniente su representación a través de tablas, que tienen a los símbolos del alfabeto de entrada encabezando las columnas y a los elementos del conjunto de estados encabezando las filas, tal como se muestra en el siguiente ejemplo.

Una alternativa para la definición de una máquina secuencial es la utilización de un grafo dirigido (*dígrafo* o simplemente grafo), donde los nodos representan los estados y los arcos dirigidos las transiciones. Esta representación es muy utilizada porque permite visualizar con mucha facilidad el comportamiento de la máquina, pero no incorpora nada nuevo a su definición formal, siendo ambas completamente equivalentes.

Ejemplo 3.1

La siguiente es una máquina de Mealy:

$$ME = (\Sigma_E, \Sigma_S, Q, f, g) \begin{cases} \Sigma_E = \{a, b, c\} \\ \Sigma_S = \{d, e, f\} \\ Q = \{p, q, r, s\} \end{cases}$$

y sus funciones de transición y salida quedan definidas por:

$$f: Q \times \Sigma_E \rightarrow Q \qquad\qquad g: Q \times \Sigma_E \rightarrow \Sigma_S$$

que son convenientemente representadas con las siguientes tablas:

***f*:**	**a** (Σ_E)	**b**	**c**
p	p	q	s
q	r	q	q
r	s	s	q
s	p	s	p

***g*:**	**a** (Σ_E)	**b**	**c**
p	d	f	f
q	f	d	e
r	d	d	f
s	d	e	e

Tabla 3.1: Funciones de transición y salida de la máquina de Mealy.

Para representar los grafos de las máquinas de Mealy, se identifican los arcos con etiquetas de tipo **e/s**, donde en cada caso $\mathbf{e} \in \mathbf{\Sigma_E}$ representa un símbolo de entrada y $\mathbf{s} \in \mathbf{\Sigma_S}$ un símbolo de salida. En la siguiente figura, se muestra el grafo que corresponde a las tablas anteriores:

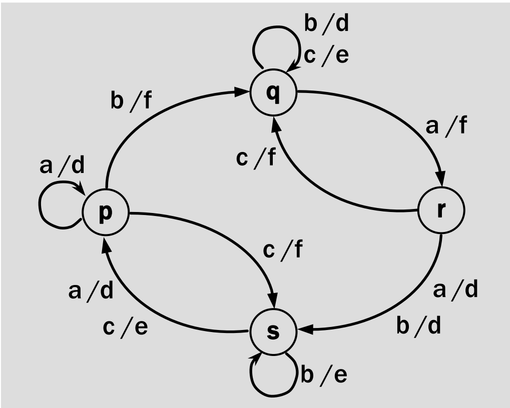


Figura 3.1: Digrafo de la Máquina de Mealy del Ejemplo 3.1.

La máquina de Mealy es traductora, lo que significa que establece una relación entre una cadena de entrada y la cadena de salida. Como ejemplo supóngase que la máquina se encuentra en un instante dado en el estado **p** y lee la cadena α**=abcab** de la cinta de entrada. En la siguiente tabla se muestran: los sucesivos estados adoptados por la máquina, la subcadena de entrada a ser leída en cada paso y la cadena de salida:

Estado	Entrada	Salida
p	abcab	λ
p	bcab	d
q	cab	df
q	ab	dfe
r	b	dfef
s	λ	dfefd

Tabla 3.2: Traducción de cadenas.

La relación entre entrada y salida puede representarse como:

Ejemplo 3.2

En la Figura 3.2, se representa un dosificador que distribuye los elementos de entrada **a** y **b** en las salidas **c**, **d** y **e** con la finalidad de ser envasados. Se desea conocer la composición de las tres salidas cuando se ingresa alternativamente la misma cantidad de productos **a** y **b** y se propone utilizar una máquina secuencial.

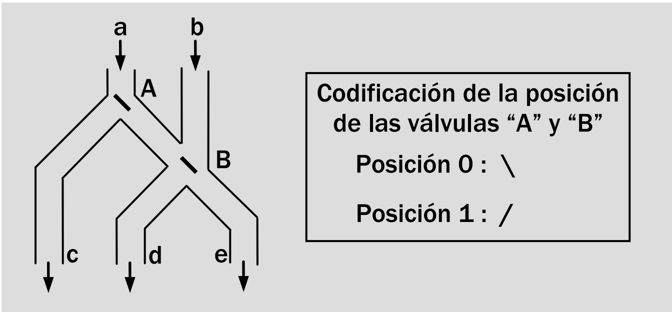


Figura 3.2: Esquema del dosificador del Ejemplo 3.2.

La operación del equipo dosificador queda determinada por las posiciones que adoptan las válvulas **A** y **B**, que son identificadas como **0** (****) y **1** (**/**), lo que da lugar a que el equipo opere en cuatro estados posibles según las posiciones de las dos válvulas (que pueden codificarse como **00**, **01**, **10** y **11**). Nótese además que, cada vez que un elemento pasa frente a una válvula, ésta cambia su posición, es decir de **0** a **1** y de **1** a **0**. Para estudiar el problema, se utiliza una máquina secuencial de Mealy, con los siguientes componentes:

$$\Sigma_E = \{a, b\} \qquad \Sigma_S = \{c, d, e\} \qquad Q = \{00, 01, 10, 11\}$$

y sus funciones de transición y de salida representadas por las tablas:

***f*:**	a	b
00	11	01
01	10	00
10	00	11
11	01	10

***g*:**	a	b
00	e	e
01	d	d
10	c	e
11	c	d

Tabla 3.3: Funciones de transición y salida del dosificador.

El dígrafo de esta máquina secuencial de Mealy es el que sigue:

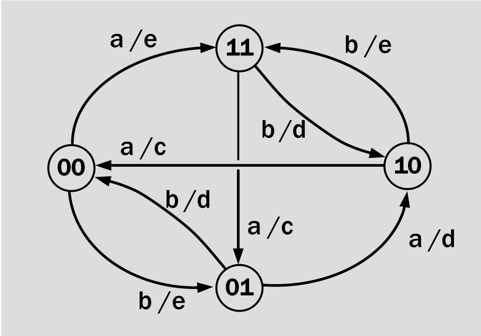


Figura 3.3: Grafo de la máquina de Mealy dosificadora.

Una vez que el dosificador ha sido modelado por una máquina secuencial, el siguiente paso es evaluar la composición de la salida para cierta composición de la entrada. Para ello, se considera que ingresa al dosificador una secuencia representada por α**=abababab** (**|α|=8**) y se estudia la evolución de la máquina con una tabla que contiene tres columnas: estado actual, secuencia pendiente de entrada y secuencia de salida obtenida. El comportamiento es representado en la Tabla 3.4.

Estado	Entrada	Salida
00	abababab	λ
11	bababab	e
10	ababab	ed
00	babab	edc
01	abab	edce
10	bab	edced
11	ab	edcede
01	b	edcedec
00	λ	edcedecd

Tabla 3.4: Operación del dosificador.

Nótese que al leerse la cadena α el dosificador ha completado un ciclo, ya que se encuentra nuevamente en el estado **00** y el próximo símbolo de entrada será nuevamente un elemento **a**. Puede deducirse entonces que por cada cadena de ocho elementos ingresada, iniciando con **a**, el dosificador realiza la clasificación mostrada en la siguiente tabla:

	Entrada	Salida **c**	Salida **d**	Salida **e**
Contenido	4a + 4b	2a	a + 2b	a + 2b
Largo	8	2	3	3
Composición	a: 50% b: 50%	a: 100%	a: 33,3 % b: 66,6 %	a: 33,3 % b: 66,6 %

Tabla 3.5: Comportamiento del dosificador ante la entrada α.

Máquina de Moore

La máquina de Moore tiene los mismos cinco componentes ya indicados para la máquina de Mealy, diferenciándose únicamente en su función de salida, ya que ésta solo depende del estado actual y no de la entrada en ese instante. Es decir que:

$$MO = (\Sigma_E, \Sigma_S, Q, f, g)$$

donde la función de transición ***f*** no cambia y en ***g*** hay una relación directa entre el estado en cada intervalo de tiempo y el símbolo de salida:

$$f: Q \times \Sigma_E \rightarrow Q \qquad g: Q \rightarrow \Sigma_S$$

Debe reconocerse que la máquina de Moore incorpora un retardo entre la entrada y la salida. En efecto, si en un instante **t** el autómata se encuentra en un estado $\mathbf{q_t \in Q}$, la salida es:

$$s_t = g(q_t)$$

y como este último estado fue a su vez alcanzado en una transición anterior,

$$q_t = f(q_{t-1}, e_{t-1})$$

se puede apreciar la relación directa entre la salida actual y la entrada en un instante anterior, que es:

$$s_t = g(f(q_{t-1}, e_{t-1}))$$

Puede demostrarse que para toda máquina de Moore hay una máquina de Mealy capaz de tener el mismo desempeño y recíprocamente. Lo primero es obvio, ya que solo basta plantear una máquina de Mealy que en cada estado prevea la misma salida para todos los símbolos de entrada. Lo opuesto, es decir, obtener una máquina de Moore a partir de una de Mealy, no es tan obvio y requiere de mayor esfuerzo.

Autómatas Finitos Deterministas (AFD)

Definición del AFD

Si a la máquina secuencial de Mealy, que había sido definida como una quíntupla, se le incorpora un estado inicial y un conjunto de estados de aceptación, estamos en presencia de un Autómata Finito Determinista (AFD). En su forma más general, el AFD es una séptupla, comienza a operar a partir de un estado inicial, tiene una conducta traductora ya que transforma las cadenas de entrada en cadenas de salida y completa su operación al terminar de leer su entrada, arribando a un estado que podrá ser de aceptación o no. Se define:

$$AFD_T = (\Sigma_E, \Sigma_S, Q, q_0, A, f, g)$$

donde

- Σ_E **:** Alfabeto de símbolos de entrada
- Σ_S **:** Alfabeto de símbolos de salida
- Q **:** Conjunto finito y 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, $f: Q \times \Sigma_E \rightarrow Q$
- *g* **:** Función de salida, $g: Q \times \Sigma_E \rightarrow \Sigma_S$

Nótese que por tener un estado inicial y ser la cadena de entrada finita, el AFD_T siempre completa su operación, de la misma forma, en una cantidad finita de tiempo y, por lo tanto, *determina un algoritmo*.

Si el AFD_T se limita a reconocer o validar cadenas, resultan innecesarios su alfabeto de salida y su correspondiente función de salida, por lo que ambas pueden ser eliminadas, dando lugar al Autómata Finito Reconocedor (AFD_R), que queda definido como una quíntupla:

$$AFD_R = (\Sigma_E, Q, q_0, A, f)$$

El significado de estos cinco componentes es el mismo que ya fue definido con anterioridad. Además, como en lo sucesivo se trabajará mayoritariamente con Autómatas Finitos Reconocedores, no será necesaria la distinción entre ambos tipos de máquinas, salvo que se indique lo contrario. Así, el AFD_R, en adelante AFD, comienza a operar a partir del estado inicial $\mathbf{q_0}$ y, al completar la lectura de la cadena de entrada, confirma su aceptación arribando a uno de los estados del conjunto **A** de estados de aceptación. Un caso particular de aceptación ocurre cuando la cadena de entrada es la palabra vacía $\boldsymbol{\lambda}$: el AFD la aceptará si su estado inicial es también un estado de aceptación: $\mathbf{q_0 \in A}$.

Una característica importante del componente ***f*** del AFD es que se trata de una función y esto es lo que lleva a definir al autómata como *determinista*. Debe recordarse que la función es un caso particular de relación entre dos conjuntos, en la cual se asocian elementos del primero (alcance) con elementos únicos del segundo (rango):

$$f : Q \times \Sigma_E \rightarrow Q$$

Como ya fue anticipado en el Capítulo 1, siempre que para todo elemento del alcance (estado actual y símbolo de entrada) se haya establecido un elemento del rango (próximo estado) la función es definida como total o completa (alcance=dominio). Por el contrario, si hay elementos del alcance que no tienen su contraparte en el rango, la función es parcial (alcance≠dominio), lo que significa que, en ciertas condiciones, el autómata no tiene definido un próximo estado de operación. Se debe interpretar que esas condiciones de operación no están previstas o corresponden a condiciones de falla, donde el autómata cesa su operación y se detiene. Para estos casos, podría incorporarse al autómata un estado de no aceptación (error), que debería ser insertado en donde corresponda para convertir la función ***f*** en completa, sin alterar el resultado obtenido. En los AFD estudiados en lo sucesivo, se admite que puedan tener sus funciones de transición parciales o completas.

Para continuar con la definición de los AFD hay otros dos aspectos que deben ser tratados, que se refieren a la lectura completa de una cadena y a la detención de la máquina.

Hay numerosas formas en las que una máquina “real” reconoce haber leído completamente una cadena. Entre otras, pueden citarse los siguientes casos: *i*) se conocen anticipadamente los largos de las cadenas a ser leídas, *ii*) los caracteres son recibidos con una periodicidad regular (caso de una entrada de comunicaciones) o *iii*) el mensaje termina con un carácter especial. Para este último caso, la tabla ASCII prevé el **ETX** (fin de texto, 03) y el **EOT** (fin de transmisión, 04), según la cadena provenga de un archivo o un protocolo de comunicaciones.

En el caso de las máquinas abstractas, se prescinde de la técnica de implementación del medio de entrada, por lo que se habla en forma genérica de una *cinta de entrada* y su correspondiente cabezal. Se admite, por lo tanto, que la máquina reconoce haber completado la lectura de una cadena sin necesidad de precisar el modo en que lo hace.

Esto significa que, a partir de una cierta cadena de entrada α, en cada intervalo de tiempo, se conoce tanto el prefijo ya leído por el AFD como también el sufijo que representa la cadena que resta ser leída. Luego, cuando la cadena ya fue leída completamente el prefijo leído es igual a α y el sufijo a ser leído es igual a λ, lo que significa que no hay nada por leer.

El segundo problema a ser tratado es el de parada de la máquina y aquí se reconocen las siguientes causas principales: *i*) la función de transición es parcial, no estando definido un próximo estado (la cadena es, por lo tanto, rechazada) y *ii*) se completó la lectura de la cadena de entrada. En esta última condición, puede ocurrir que: a) la máquina se encuentra en un estado de aceptación ($\mathbf{q \in A}$) y, por eso, la cadena es aceptada o reconocida, y b) se encuentra en cualquier otro estado ($\mathbf{q \notin A}$), lo que indica que la cadena es desconocida o rechazada.

Por lo expuesto, se admite en lo sucesivo que todo estado $\mathbf{q \in A}$ es un estado de aceptación (se dibujará con doble círculo en el dígrafo) y todo estado $\mathbf{p \in (Q - A)}$ es un estado de no aceptación o rechazo.

Configuración o descripción instantánea

Se define como configuración o descripción instantánea $\mathbf{K_t}$ de un autómata finito en un intervalo de tiempo t al par ordenado:

$$K_t = (q, \beta) \quad ; \quad \text{con } q \in Q,\ \beta \in \Sigma_E^*$$

donde q representa el estado en el que se encuentra y $\boldsymbol{\beta}$ el sufijo o subcadena de entrada que está pendiente de ser leída. A partir de esta definición, y ante una cadena de entrada $\boldsymbol{\alpha}$ a ser procesada, se puede reconocer la configuración inicial como:

$$K_0 = (q_0, \alpha)$$

De igual forma, se define la configuración de aceptación como:

$$K_n = (q_n, \lambda) \quad ; \quad \text{donde } q_n \in A,\ n=|\alpha|$$

El tránsito de una configuración a otra es denominado movimiento, es decir que si existe la transición **q=f(p, a),** el movimiento del estado **p** al **q** leyendo el símbolo de entrada a, puede representarse:

$$(p, a\beta) \vdash (q, \beta)$$

El movimiento desde la configuración inicial a la final es representado:

$$(q_0, \alpha) \vdash^* (q_n, \lambda)$$

donde $\vdash^*$ equivale a una cantidad finita de movimientos que es igual al largo **|α|** de la cadena de entrada. Se reitera que se utiliza λ para indicar que en la cinta de entrada no ha quedado nada por leer.

El comportamiento del autómata ante cierta cadena de entrada queda convenientemente representado por un *árbol de configuraciones* o de descripciones instantáneas, y como alternativa puede también utilizarse un esquema denominado *plano de estados–entradas*. Ambas representaciones muestran las sucesivas descripciones instantáneas durante la operación del autómata y son utilizadas en los siguientes ejemplos.

Ejemplo 3.3

El AFD representado en la Figura 3.4, está destinado a reconocer cadenas que respondan a la forma general **α=(0+1)*1000**.

Se pide identificar su definición formal y comprobar la aceptación de una cadena que responda a esa expresión regular.

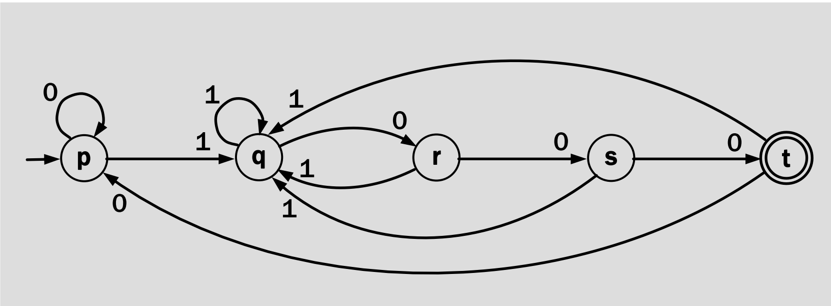


Figura 3.4: Grafo del AFD del Ejemplo 3.3.

A partir de la inspección del grafo, se deduce su definición formal:

AFD **= (Σ_E, Q, q_0, A, *f*)**

Σ_E = {0, 1}
Q = {p, q, r, s, t}
q_0 = p (indicado con →)
A = {t} (indicado con ◎)

***f*:**	**0**	**1**
→ p	**p**	**q**
q	**r**	**q**
r	**s**	**q**
s	**t**	**q**
*** t**	**p**	**q**

Tabla 3.6: Función ***f*** del AFD del Ejemplo 3.3.

Una vez que el autómata ha sido definido es necesario comprobar su correcto comportamiento[1] y para ello se deben hacer múltiples pruebas con cadenas cuidadosamente seleccionadas, utilizando los árboles de descripciones instantáneas o planos de estados-entradas ya anticipados.

A título de ejemplo, y con el fin de comprobar el funcionamiento del AFD estudiado, se recurre a dos cadenas tales como la **β=1011000** y **δ=1011100**. La primera cadena responde a la expresión regular definida **(α)** y la segunda cadena propuesta no cumple con la condición requerida de contener un sufijo **1000**.

Se representan en la Figura 3.5 los dos árboles de descripciones instantáneas que corresponden a la operación del AFD ante la lectura de las cadenas indicadas. Como puede observarse, la primera cadena β es aceptada al completarse su lectura por haber alcanzado el AFD un estado

[1] El correcto funcionamiento puede demostrarse por inducción matemática sobre el largo de la cadena de entrada, lo que está fuera del alcance dado a este texto.

de aceptación, mientras que la segunda cadena **δ** es rechazada. Además, nótese que los árboles son lineales por tratarse precisamente de un AFD, en el cual para cada estado y símbolo leído existe una única transición posible.

Los movimientos representados en los árboles de configuraciones pueden resumirse en:

$$\text{Lectura de cadena } \beta\text{:} \quad (p,\ 1011000) \vdash^{*} (t,\ \lambda)$$
$$\text{Lectura de cadena } \delta\text{:} \quad (p,\ 1011100) \vdash^{*} (s,\ \lambda)$$

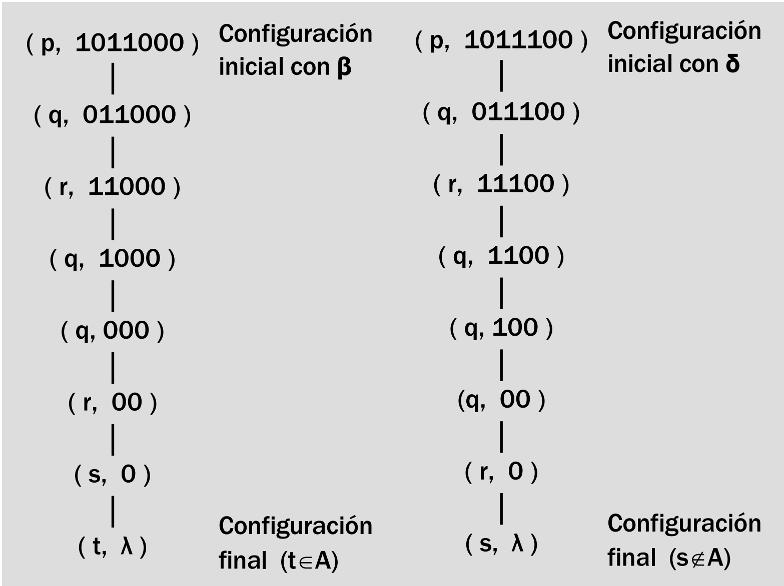


Figura 3.5: Árboles de descripciones instantáneas del Ejemplo 3.3.

A continuación, se presenta en la Figura 3.6, el plano de estados-entradas que corresponde a la aceptación de la cadena **β**:

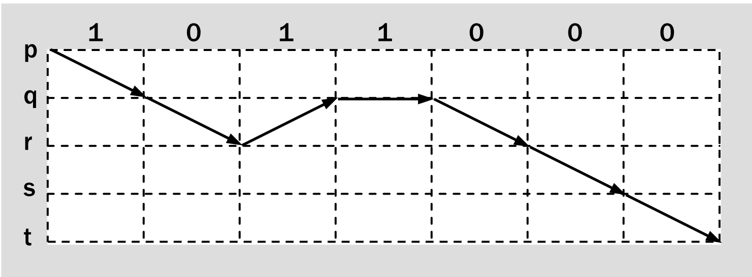


Figura 3.6: Representación del plano estados-entradas.

Ejemplo 3.4

Representar una máquina expendedora de golosinas con un autómata finito, donde sus estados están asociados al importe acumulado por las monedas recibidas. El alfabeto de entrada está formado por los diferentes valores de monedas reconocidas (5 y 10 centavos) y los pulsadores de selección de golosinas (<C>hicle y <P>astillas). Las golosinas disponibles cuestan 15 centavos (chicles) y 20 centavos (pastillas). La máquina debe devolver las monedas ingresadas en exceso y dar vuelto, según corresponda. Además, al retirarse el producto la máquina, vuelve a su estado inicial, de esta manera queda disponible para la próxima operación.

Comentarios:

Este expendedor presenta la característica de que si el importe es insuficiente para la golosina seleccionada no se entrega nada, quedando a la espera de más monedas, y si el importe supera el valor de la golosina seleccionada se devuelve el importe excedente. Así, una vez alcanzados los estados $\mathbf{q_{15}}$ y $\mathbf{q_{20}}$ queda a la espera de la orden para entregar la golosina y cuando corresponde devuelve el importe excedente. Luego, un sensor detecta que se han retirado las golosinas y lo informa al sistema con el estímulo **R** que lo lleva nuevamente a la condición de inicio de operación.

Solución

El expendedor de golosinas es representado por un autómata finito traductor que responde a la siguiente definición formal:

$$AFD_T = (\Sigma_E,\ \Sigma_S,\ Q,\ q_0,\ A,\ f,\ g)$$

donde:

$\Sigma_E = \{5, 10, C, P, R\}$; $\Sigma_S = \{5, 10, n\}$; (n = s/respuesta o "no acción")

$Q = \{s_0, s_5, s_{10}, s_{15}, s_{20}, s_C, s_P\}$; $q_0 = s_0$; $A = \{s_C, s_P\}$

Las funciones de transición ***f*** y de salida ***g*** son:

f:	5	10	C	P	R
→s_0	s_5	s_{10}	s_0	s_0	s_0
s_5	s_{10}	s_{15}	s_5	s_5	s_5
s_{10}	s_{15}	s_{20}	s_{10}	s_{10}	s_{10}
s_{15}	s_{20}	s_{20}	s_C	s_{15}	s_{15}
s_{20}	s_{20}	s_{20}	s_C	s_P	s_{20}
*s_C	s_C	s_C	s_C	s_C	s_0
*s_P	s_P	s_P	s_P	s_P	s_0

g:	5	10	C	P	R
→s_0	*n*	*n*	*n*	*n*	*n*
s_5	*n*	*n*	*n*	*n*	*n*
s_{10}	*n*	*n*	*n*	*n*	*n*
s_{15}	*n*	5	*n*	*n*	*n*
s_{20}	5	10	5	*n*	*n*
*s_C	5	10	*n*	*n*	*n*
*s_P	5	10	*n*	*n*	*n*

Tabla 3.7: Funciones de transición y salida del expendedor de golosinas.

A partir de la definición formal del autómata, se representa su grafo en la Figura 3.7:

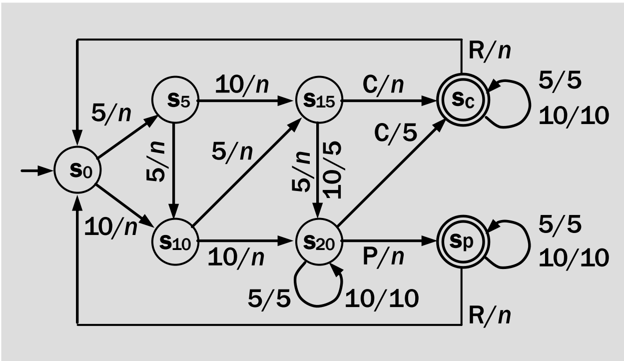


Figura 3.7: Funciones de transición y salida del expendedor de golosinas.

Nótese que, por razones de claridad, no se han representado en el grafo de la Figura 3.7 las transiciones sobre los mismos estados cuando se ingresan los símbolos **C**, **P** y **R**.

En el caso anterior, la operación de la máquina puede representarse con el árbol de descripciones instantáneas y con el plano de estados-entradas. En este caso, se lo hace expendiendo pastillas, tal como se muestra en la Figura 3.8:

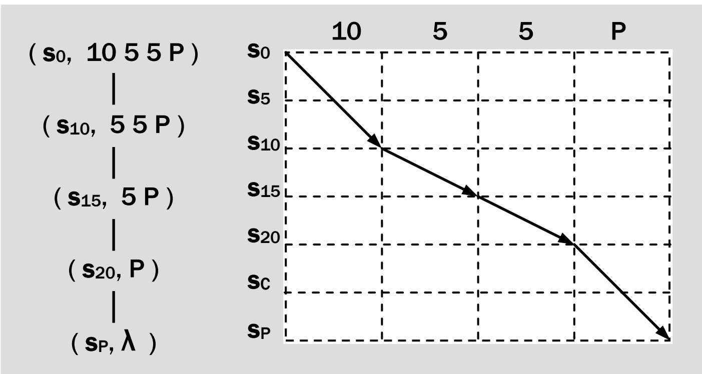


Figura 3.8: Árbol de descripciones instantáneas y plano estados-entradas de la operación del expendedor de golosinas.

Extensión al tratamiento de palabras

La función de transición ***f*** define el comportamiento de un AFD en cada estado y ante la entrada de cada posible símbolo, quedando expresado este comportamiento como la transición a otro estado. Esta función puede ser extendida para describir lo que ocurre a partir de cada estado si en lugar de recibir símbolos aislados se recibe una secuencia concatenada de los símbolos de entrada. La noción de función de transición extendida $f^e: Q \times \Sigma_E^* \to Q$, para operar sobre cadenas, puede inducirse a partir de un razonamiento muy simple, considerando que la cadena **α=abcde** puede ser expresada como **α=aβ**, donde **β=bcde**:

$$f^e(q, \alpha) = f^e(q, a\beta) = f^e(f(q, a), \beta)$$

Se obtiene un proceso recursivo que finaliza al completarse la lectura de **α**, situación en donde ocurre que **$f^e(q, \lambda)=q$**. Así, se tiene que:

$$f^e(q,\alpha) = f(f(f(f(f(q,a),b),c),d),e)$$

El proceso de reconocimiento de cadenas del Ejemplo 3.3 puede ser expresado mediante la función de transición extendida según se muestra:

$$(p, 1011000) \vdash^* (t, \lambda) \Rightarrow f^e(p, 1011000) = t$$
$$(p, 1011100) \vdash^* (s,\lambda) \Rightarrow f^e(p, 1011100) = s$$

Este concepto también puede aplicarse para extender la función de salida de las máquinas secuenciales de Mealy o Moore y del autómata finito traductor. En el caso de la máquina de Mealy del Ejemplo 3.1, la función de salida extendida determina que:

$$g^e(p, abcab) = dfefd$$

quedando establecida una relación entre las cadenas de entrada y salida.

Aceptación de cadenas y otros conceptos asociados a los AFD

Aceptación de palabras y lenguajes

Si en el movimiento desde la configuración inicial hasta la final, representado por **$(q_0, \alpha) \vdash^* (q_n, \lambda)$**, se verifica que **$q_n \in A$**, se dice que la cadena **α** ha sido *reconocida* o *aceptada*. El AFD reconocedor llega a la conclusión de aceptar o rechazar la cadena de entrada luego de efectuar un número finito de transiciones o movimientos a partir de su configuración inicial. Es decir que la aceptación de una cadena por parte de un AFD implica dos condiciones simultáneas: *i*) la completa lectura de la cadena de entrada **α** y *ii*) el arribo a un estado de aceptación. Por el contrario, si **$q_n \notin A$** la cadena **α** habrá sido rechazada.

A partir de lo expuesto, se dice que *un lenguaje **L** es aceptado por un autómata finito **M*** si todas las palabras que lo componen conducen el autómata a estados de aceptación. En símbolos:

$$L(M) = \{\alpha \ / \ \alpha \in \Sigma_E^* \text{ y } f^e(q_0, \alpha) \in A\}$$

o en forma equivalente:

$$L(M) = \{\alpha \ / \ \alpha \in \Sigma_E^* \text{ y } (q_0, \alpha) \vdash^* (q_n, \lambda) \text{ y } q_n \in A\}$$

Accesibilidad entre estados

En un AFD, se dice que *un estado **p** es accesible desde otro estado **q***, y se representa **pAq**, si existe una palabra de símbolos de entrada que haga transitar el autómata desde el estado **q** hasta el estado **p**. En símbolos:

$$pAq \leftrightarrow \exists \alpha \in \Sigma_E^* / \ f^e(q, \alpha)=p, \quad \text{con } p, q \in Q$$

Nótese que todos los estados inaccesibles desde el estado inicial **q_0** pueden ser eliminados de un autómata finito ya que no afectan su comportamiento, en cuanto a reconocimiento de cadenas se refiere.

Autómatas conexos

Si todos los estados de un autómata son accesibles desde su estado inicial se dice que el mismo es *conexo*. En símbolos:

$$\textbf{Un AFD es conexo} \leftrightarrow \forall p_i \in Q\text{: } p_i A q_0$$

Equivalencia de estados y autómatas

Equivalencia entre estados

Dos estados p,q∈Q de un AFD se dice que son equivalentes, lo que es representado por pEq, si desde cualquiera de ellos se aceptan y rechazan exactamente las mismas cadenas de símbolos de entrada.

En símbolos:

$$pEq \leftrightarrow [\forall \alpha \in \Sigma_E^*\text{: } f^e(p, \alpha) \in A \leftrightarrow f^e(q, \alpha) \in A]$$

Se reconoce entonces que entre dos elementos p y q del conjunto de estados posibles **Q**, se ha establecido una relación de equivalencia **E**, que satisface las propiedades reflexiva, simétrica y transitiva en **Q**.

Nótese que, según la relación de equivalencia presentada, es imposible distinguir entre dos estados equivalentes, lo que lleva a denominarlos indistinguibles. Además, si los estados **p** y **q** son equivalentes y **|A|>1**, puede ocurrir que los estados de aceptación o rechazo alcanzables a partir de cierta cadena α, no sean necesariamente los mismos.

Equivalencia de longitud "k" entre estados

Dos estados **p, q∈Q** de un AFD son equivalentes de longitud k, lo que se representa con **pEkq**, si son equivalentes para todas las cadenas de largo menor o igual a k. Esto puede expresarse en símbolos:

$$pE_kq \leftrightarrow [\forall \alpha \in \Sigma_E^*\text{: } |\alpha| \leq k \text{ y } f^e(p, \alpha) \in A \leftrightarrow f^e(q, \alpha) \in A]$$

Equivalencia entre autómatas

Dos AFD denominados $\mathbf{A_1}$ y $\mathbf{A_2}$, se dice que son equivalentes y se representa $\mathbf{A_1EA_2}$, si ambos aceptan las mismas palabras o sentencias. De aquí se deduce fácilmente que dos AFD son equivalentes si lo son sus estados iniciales. Lo anterior implica además, que dos AFD son equivalentes si y solo si aceptan el mismo lenguaje. En símbolos:

$$A_1EA_2 \leftrightarrow L(A_1)=L(A_2)$$

Conjunto cociente

Como toda relación de equivalencia, la relación E definida sobre el conjunto de estados posibles **Q**, induce en el mismo una partición. Todos los elementos de **Q** relacionados con cierto estado **p∈Q,** constituyen un subconjunto de **Q** que es denominado clase de equivalencia de **p**.

Se denomina conjunto cociente a esa partición, conjunto que tiene por elementos a todas las clases de equivalencia o subconjuntos de estados equivalentes entre sí que puedan distinguirse en el conjunto de estados **Q**. Por ser la base de este agrupamiento, la relación de equivalencia entre estados E, el conjunto cociente se denota **Q/E**.

La definición de equivalencia entre estados no proporciona un procedimiento para saber si dos estados **p,q∈Q** son equivalentes, ya que hay infinitas sentencias $\alpha \in \Sigma_E^*$, aún en caso de que el alfabeto $\mathbf{\Sigma_E}$ tenga un solo símbolo. Por el contrario, con el concepto de equivalencia de longitud **k** entre estados, es teóricamente posible saber si dos estados **p,q∈Q** son k-equivalentes ya que hay un número finito de sentencias $\alpha \in \Sigma_E^*$ con $|\alpha| \leq k$. Sin embargo, se trata de un proceso enormemente laborioso en la mayoría de los casos.

Para superar esta dificultad, se recurre a un procedimiento práctico incremental que comienza con un particionamiento o conjunto cociente inicial de **Q**, que es sucesivamente revisado con cadenas progresivamente más largas, a partir de **k=1** y hasta que las clases de equivalencia de **Q** se estabilicen en cantidad y contenido, no sufriendo cambios para cadenas de mayor longitud.

En el caso de las máquinas abstractas reconocedoras, como son los AFD estudiados, el conjunto cociente inicial (con **k=0**, lo que significa tener solo en cuenta la cadena λ, única con ese largo) es:

$$Q/E_0 = P_1^0 \cup P_2^0,\text{ donde } P_1^0 = Q - A \quad y \quad P_2^0 = A$$

Así, el conjunto cociente inicial tiene dos elementos: un subconjunto que contiene a los estados de aceptación **($P_2$0)** y otro subconjunto con los estados restantes, de no aceptación **($P_1$0)**.

Los siguientes conjuntos cocientes **Q/E_{k+1}** se definen dividiendo las clases **P_i^k** de **Q/E_k** las veces que sea necesario hasta que, para todo símbolo $a \in \Sigma_E$ y para todo par de estados p, q de las clases **P_i^k**, queden definidos movimientos que conduzcan a elementos de las mismas clases de **Q/E_{k+1}**. Esto se repite hasta obtener **$Q/E_{k+1} = Q/E_k$.**

Lo anterior significa que, para todo símbolo de entrada **$a \in \Sigma_E$** y para todo par de estados **$p,q \in P_i^k$**, es decir que **pE_kq**, la existencia de una condición $f(p,a) \in P_i^k$ y $f(q,a) \notin P_i^k$ implicará que el estado **p** no es equivalente de longitud **k+1** al **q**, por ello que deben separarse en distintas clases de equivalencia en el siguiente conjunto cociente:

$$P_i^{k+1} = P_i^k - \{q\},\ \text{en donde } q \in P_{i+1}^{k+1} \text{ y } P_{i+1}^{k+1} \in Q/E_{k+1}$$

Minimización de autómatas

Los autómatas son construidos para representar con claridad las condiciones de operación del objeto estudiado, tal como quedó en evidencia al utilizarse una máquina de Mealy para representar un dosificador (Ejemplo 3.2) o un autómata finito para representar el comportamiento del expendedor de golosinas (Ejemplo 3.4). Se trata de construir modelos, y una relación directa con el objeto representado facilita su definición y contribuye a evitar omisiones y errores. Sin embargo, esto no asegura que se haya obtenido el autómata más eficiente, lo que implica que es muy probable que el modelo tenga más estados que los estrictamente necesarios para operar correctamente. Estos estados fueron necesarios para asegurar un proceso de modelado apropiado, pero son excesivos a la hora de implementarlos a través de código en algún lenguaje de programación. Surge así el concepto de *autómata mínimo*, que se define como aquel que cumple correctamente su función con la menor cantidad posible de estados. Viendo al problema desde otro ángulo, todo autómata que sea conexo y opere correctamente con más estados de los necesarios, debe tener en su definición estados equivalentes o indistinguibles.

Esto significa que tal autómata debe ser minimizado y para ello se recurre al concepto de conjunto cociente visto con anterioridad, es decir, a la partición del conjunto de estados Q producida por la relación de equivalencia E, que tiene por elementos a las clases de equivalencia, donde se agrupan todos los estados equivalentes o indistinguibles entre sí.

El autómata mínimo equivalente a uno dado, tendrá como estados a cada una de las partes o clases de estados equivalentes del autómata original (es decir, su conjunto de estados será Q/E), aquella clase que contenga el estado inicial original será su estado inicial, las que contengan algún estado de aceptación del autómata original serán sus estados de aceptación; su función de transición se construirá en base a la del autómata original, como se muestra en los siguientes ejemplos.

Ejemplo 3.5

Se desea reconocer cadenas, que responden al patrón **$(a+b)^*ba$** y para ello se propone el *AFD* que se muestra en la figura 3.9:

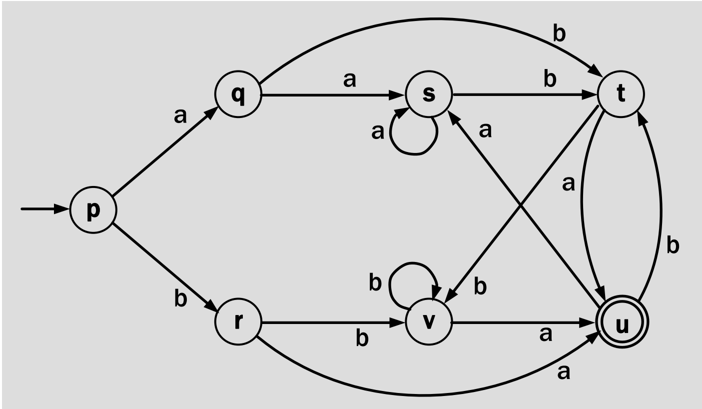


Figura 3.9: Dígrafo del AFD del Ejemplo 3.5.

Nótese que este AFD fue construido con el objetivo de reconocer todas las posibles combinaciones en la secuencia de caracteres de una cadena α que tenga el patrón solicitado. El interrogante es si algún AFD con menor cantidad de estados puede cumplir esta misma función y con este fin se intentará minimizarlo. Para ello, se comienza por definir la tabla de la función de transición, verificar que el autómata es conexo y luego determinar el conjunto cociente de sus estados.

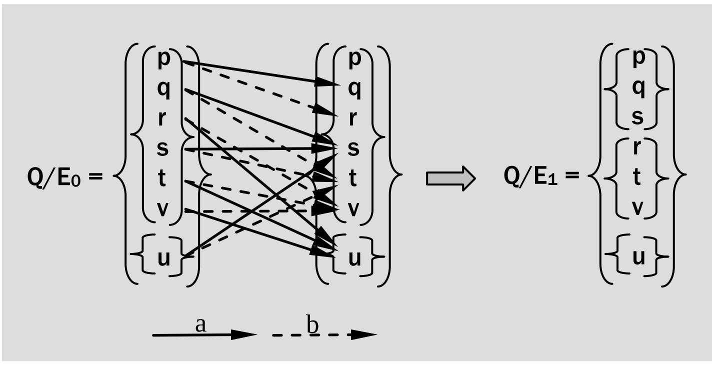
***f*:**	a	b
→ p	q	r
q	s	t
r	u	v
s	s	t
t	u	v
* u	s	t
v	u	v

Tabla 3.8: Función ***f*** de transición del Ejemplo 3.5.

La primera partición del conjunto de estados o conjunto cociente inicial (**k=0**) se obtiene separando al estado final del resto. Se obtiene así:

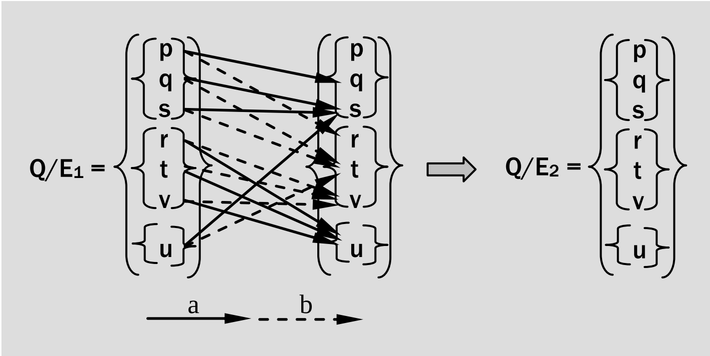
$$Q/E_0 = \{\{p, q, r, s, t, v\}, \{u\}\}$$

Para esclarecer el proceso de redistribución progresiva de los estados equivalentes en los subconjuntos existentes y la incorporación de otros subconjuntos nuevos, se recurre al esquema que se muestra a continuación:

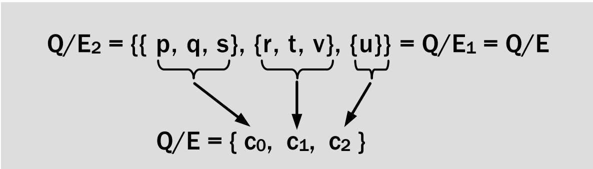


Obsérvese que el esquema corresponde a **k=1** y permite comprobar que los estados **p**, **q** y **s** tienen el mismo comportamiento, permaneciendo en la clase en que se encontraban para todos los símbolos del alfabeto de entrada. Por el contrario, los estados **r**, **t** y **v** tienen una conducta común pero diferente a la anterior: ante una entrada **a** transitan a otra clase mientras que con una entrada **b** permanecen en la misma clase. Esto significa que los seis estados de la primera clase no son todos equivalentes con **k=1**, por lo que deben ser agrupados en dos subconjuntos diferentes. Por último, la segunda clase de **Q/E₀** tiene un solo estado **u**, por lo que no requiere ningún análisis. A partir del estudio realizado, se define el nuevo conjunto cociente **Q/E$_1$**, que corresponde a **k=1** e incluye una nueva partición de estados, tal como se muestra en el mismo esquema anterior.

El siguiente paso es repetir el procedimiento cuando se lee un nuevo símbolo, lo que implica buscar las clases de equivalencia para **k=2**. El esquema es ahora el siguiente:



Como puede comprobarse, al leerse un símbolo adicional (**k=2**) los estados agrupados en cada subconjunto mantienen su equivalencia entre sí, lo que significa que el conjunto cociente se ha estabilizado y no sufrirá variantes para cadenas de mayor longitud. Esto puede expresarse:



habiéndose asignado nuevos nombres a los conjuntos de estados resultantes.

El *AFD* mínimo tiene la definición formal y dígrafo siguiente:

$$AFD = (\Sigma_E, Q, q_o, A, f)$$

donde $\Sigma_E = \{a, b\}$, $Q = \{c_0, c_1, c_2\}$, $q_0 = c_0$, $A = \{c_2\}$ y f se representa en la Tabla 3.9.

f:	a	b
→ c_0	c_0	c_1
c_1	c_2	c_1
* c_2	c_0	c_1

Tabla 3.9: Función de transición.

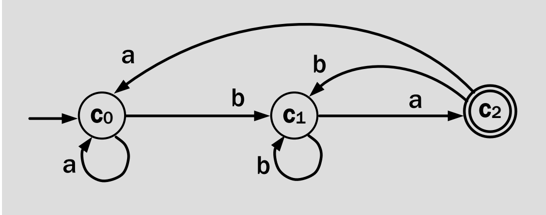


Figura 3.10: Dígrafo del AFD minimizado.

//