Autómatas, Gramáticas y Lenguajes

Unidad 2

Máquinas secuenciales y computacionales

Tecnología en Desarrollo de Software

IU Digital de Antioquia
INSTITUCIÓN UNIVERSITARIA DIGITAL DE ANTIOQUIA

UNIDAD 2: Máquinas secuenciales y computacionales

Bienvenido a la segunda unidad del curso de Autómatas, gramáticas y lenguajes. A estas alturas, debes tener claro que los autómatas son algoritmos representados mediante grafos. Los que aprendiste a crear en la unidad 1, se usan para reconocimiento de cadenas de texto y aceptarlas solamente si cumplen con las reglas gramaticales estipuladas. En la segunda unidad, podrás ver que los autómatas, no solamente sirven para reconocimiento de expresiones gramaticales, sino que permiten crear algoritmos lógicos y matemáticos para una infinidad de aplicaciones. Estos autómatas, llamados máquinas virtuales, son la esencia misma de la computación.

Introducción a la unidad 2

En la primera parte de esta unidad, verás las máquinas de Mealy y las máquinas de Moore, conocidas como máquinas de estado finito, ya que funcionan con base en autómatas finitos. También se conocen como máquinas secuenciales, debido a que resuelven algoritmos basados en secuencias de estados. Estas máquinas secuenciales son de mucho uso en reconocimiento de contraseñas, en automatización de procesos industriales y en robótica principalmente.

En la segunda parte de esta unidad, verás las máquinas de Turing, que, usando una cinta en lugar de una pila, puede correr cualquier algoritmo que se le pida. Recuerda que un algoritmo es un conjunto de pasos finitos que se detienen cuando encuentran la solución a un problema. Por lo tanto, un problema que llegue a un bucle infinito por alguna indeterminación y nunca se detenga, no se podría resolver con una máquina de Turing. Estos son los llamados, problemas indecidibles. Pero mientras exista un algoritmo que resuelva el problema, la máquina de Turing podrá ejecutarlo.

Objetivos de aprendizaje de la Unidad 1

1. Construir algoritmos que reconozcan cadenas secuenciales que generen respuestas con base, tanto en la secuencia de estados anteriores, como en el estado actual.
2. Diseñar autómatas capaces de simular cualquier algoritmo de computación, manipulando símbolos sobre una cinta de acuerdo a unas reglas establecidas para la comprensión de los fundamentos de las funciones dentro de un procesador.

Cronograma de actividades de la unidad

Actividad de aprendizaje*	Evidencia de aprendizaje**	Semana***
AA3. Máquinas de Mealy y Máquinas de Moore	EA3. Máquinas secuenciales. Resolución de problemas.	Semanas 6 y 7
AA4. Actividad de cierre.	EA4. Máquinas de Turing. Resolución de problemas.	Semana 8
Total		

Desarrollo temático de la unidad 2

1. Máquinas de Mealy y Máquinas de Moore.

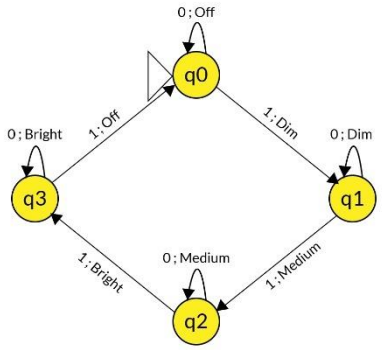
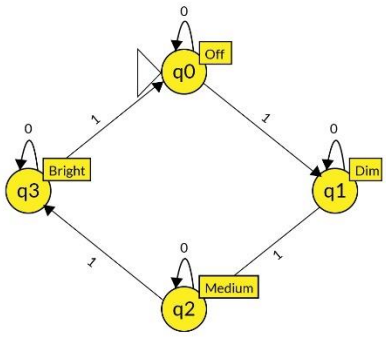
Las máquinas son un tipo de autómatas de estados finitos que generan una salida a partir de una entrada secuencial, basándose en su estado actual. Esto significa que el cambio de un estado a otro, depende, no solamente de la última señal de entrada recibida, sino de la señal o señales anteriores y del orden en que se recibieron. Una señal puede ser un voltaje causado por un pulsador, un interruptor, un sensor de proximidad, un sensor de luz, etc.

Estas máquinas no tienen estado final, quiere decir que pueden funcionar indefinidamente, siempre a la espera de una señal de entrada.

1.1. Máquinas secuenciales básicas:

El siguiente ejemplo ilustra el funcionamiento de ambas máquinas. Las de Mealy y las de Moore. Considera los siguientes artefactos. Ambos casos trabajan igual:

Una lámpara de noche puede tener 4 estados de iluminación: ● Apagado ● Tenue (Dim) ● Medio (Medium) ● Brillante (Bright)	Un limpia parabrisas de un auto puede tener 4 estados de desempeño: ● Apagado ● Intermitente (Lento) ● Medio (Medium) ● Rápido (Fast)
Ambos dispositivos cambian de estado cada vez que se oprime el pulsador. El pulsador puede estar en posición 1 / 0	

<table>
<tr><th>MÁQUINA DE MEALY</th><th>MÁQUINA DE MOORE</th></tr>
<tr><td>Figura 1. Funcionamiento Máquina Mealy
 
</td><td>Figura 2. Funcionamiento Máquina Moore
 
</td></tr>
<tr><td>Máquina de Mealy: Tanto la señal de entrada, como el nombre del siguiente estado, se indican en la transición. Si la lámpara de cuatro estados se encuentra en q3, ésta seguirá en Bright mientras el pulsador esté en posición 0.

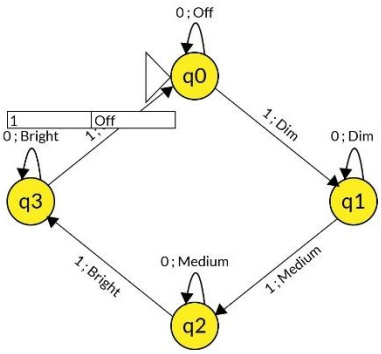
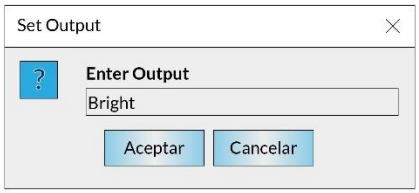
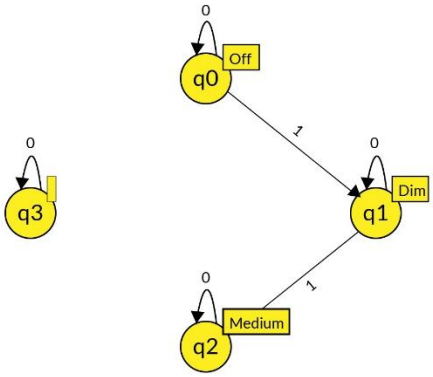
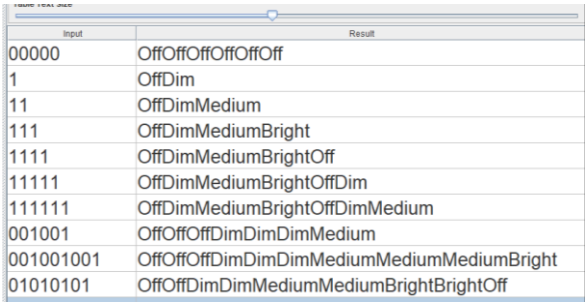
Al cambiar el pulsador a 1, en la transición de q3 a q0, el estado cambia de Bright a Off, como se ve en la Figura 3.</td><td>Figura 3. Estado de Bright a Off
 
</td></tr>
<tr><td>Máquina de Moore: La transición solamente tiene la señal de entrada. El nombre del estado se marca en el mismo estado en el momento de crearlo, como se muestra a continuación.

Figura 4. Transición de la señal de entrada
 
</td><td>Figura 5. Estado q3 con el nombre Bright
 
</td></tr>
</table>

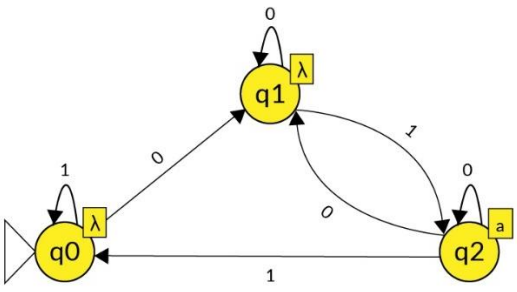
En este caso, se está creando el estado q3 con el nombre de Bright, como se ve en la Figura 5.	
Simulación en el JFLAP: Los resultados de la máquina de Mealy y la de Moore en el circuito anterior, son exactamente iguales (ver Figrura 6). Cada vez que recibe un 1, cambia al siguiente estado. Pero si lee 0, permanece en el estado actual. Toda máquina de Mealy se puede convertir en máquina de Moore y viceversa.	*Figura 6*. Resultados máquina Mealy y Moore 

Input	Result
00000	OffOffOffOffOffOff
1	OffDim
11	OffDimMedium
111	OffDimMediumBright
1111	OffDimMediumBrightOff
11111	OffDimMediumBrightOffDim
111111	OffDimMediumBrightOffDimMedium
001001	OffOffOffDimDimDimMedium
001001001	OffOffOffDimDimDimMediumMediumMediumBright
01010101	OffOffDimDimMediumMediumBrightBrightOff

1.2. Reconocimiento de secuencias:

Esta aplicación es muy útil para aceptar o rechazar contraseñas. En la Figura 7, se explica un caso con un ejemplo. Se pide una máquina de Moore que tiene un alfabeto de entrada Σ= (0, 1) y que genere una letra “a” solamente cuando encuentre un 1 antecedido de un 0. Es decir, cada sub-cadena 01 genera una “a”.

Figura 7. Ejemplo de reconocimiento de secuencias

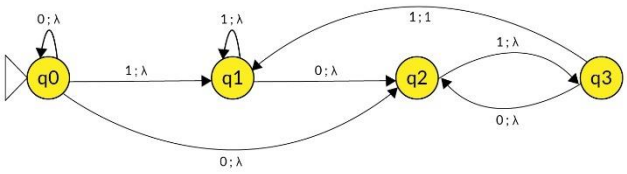
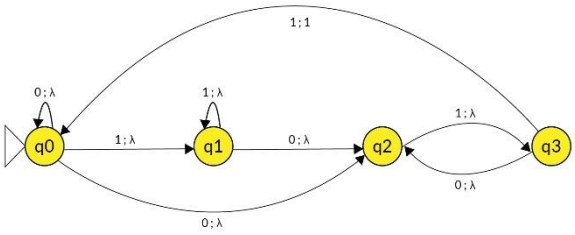


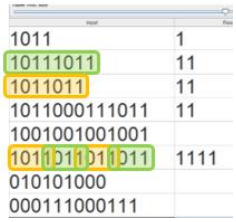
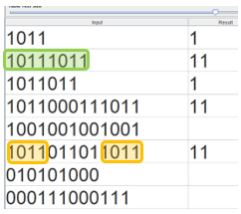
1011	a
1001	a
1111	
0110	a
00000	
1111000	
101100101001	aaaa
1111011101111101110	aaa
0101010101010101	aaaaaaaaa

Las señales de entrada son el mismo alfabeto del lenguaje Σ. Observa que la "a" está en el estado q2. La única forma de llegar a q2 es después de leer la secuencia 01. Como ejercicio debes hacer la máquina de Mealy equivalente y comprobarla con las mismas cadenas de entrada.

1.3. Contador de secuencias con máquina de Mealy:

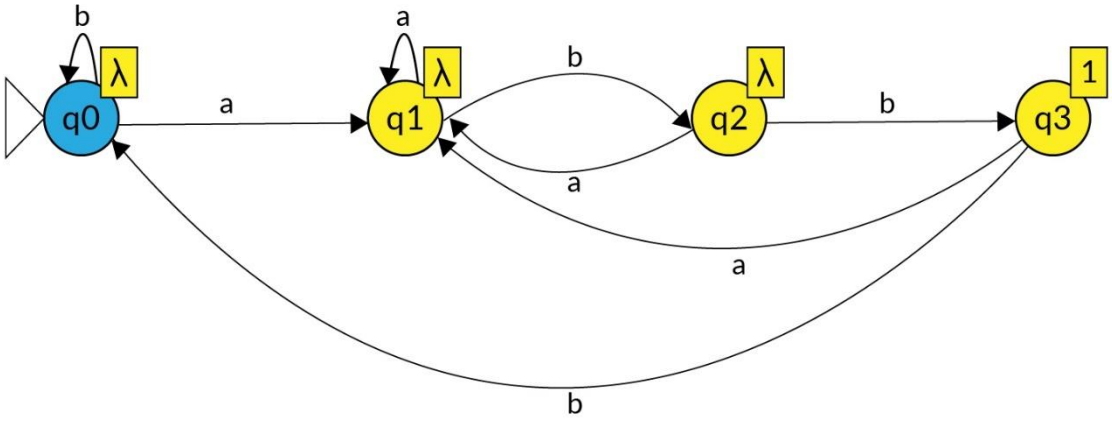
Una operación lógica es contar cuántas veces ocurre un evento o aparece una sub-cadena en una cadena. Mira el siguiente ejemplo para entender cómo funciona. Diseña una máquina de Mealy que tenga un alfabeto de entrada Σ= (0, 1), y cuenta cuántas veces detecta la cadena 1011 en una secuencia de entrada. Debe generar un 1 por cada secuencia completa que detecte.

Solución 1: Puede aceptar secuencias traslapadas como esta, 101**1**011, donde hay 2 cadenas completas. La cadena 1011 está dos veces, aunque hay un dígito que pertenece a ambas. El resultado será 11, como se muestra en la Figura 8.	**Solución 2:** Solamente acepta cadenas independientes. No puede tener caracteres traslapados. Debe leer 10111011 para reconocer dos cadenas, como se muestra en la Figura 9.
Figura 8. Secuencias traslapadas	*Figura 9.* Reconocimiento de dos cadenas
	
La única transición que termina en 1 es la que va de q3 a q1 y solamente se puede llegar allí, después de leer 1011. Ver Figura 8. La secuencia 1011 puede iniciar en q0, en q1, en q2 o en q3 como en la Figura 10.	La única transición que termina en 1 es la que va de q3 a q0. y solamente se puede llegar allí, después de leer 1011. Ver Ilustración 9. La secuencia 1011 puede iniciar en q0, en q1 o en q2. Ver Figura 11.
Figura 10. Inicio de la secuencia 1011	*Figura 11.* Inicio de la secuencia 1011

1.4. Contador de secuencias con máquina de Moore:

Diseña una máquina de Moore con alfabeto de entrada Σ= (a, b), de modo que genere un 1 cada vez que detecte la secuencia abb.



Solución: En el autómata de la Figura 12 se observa que solamente hay un 1 en la salida de q3. Los demás estados tienen salida = λ.

La cadena “abb” puede iniciar en q0, en q1 o en q3 como lo ves en la Figura 13.

Figura 13. Cadena “abb”

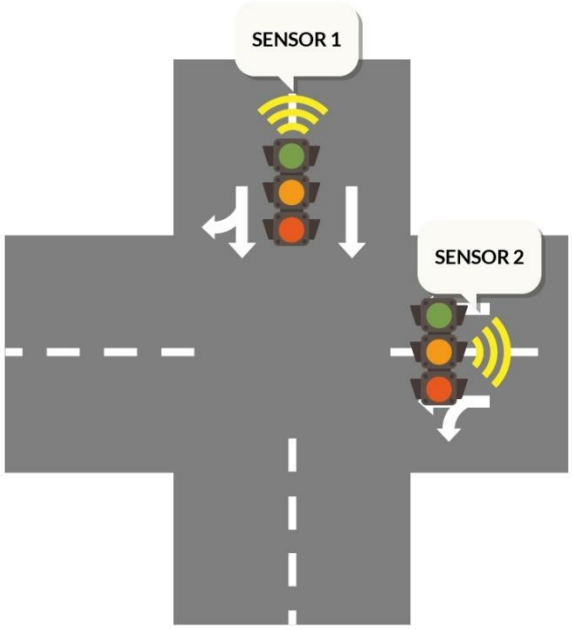
ab	
abb	1
aaabbbb	1
bbbabab	
bbabbabbabbabb	1111

1.5. Automatización de equipos con máquinas secuenciales:

Una de las principales aplicaciones de las máquinas secuénciales, es la automatización de equipos industriales. Primero se usaron sistemas electro-mecánicos, luego circuitos electrónicos con transistores, circuitos integrados, etc., y ahora, los microprocesadores ayudan a simplificar y perfeccionar estas operaciones, como podrás ver en el siguiente caso.

Cruce vial de dos semáforos con sensores. Diseña una máquina de Moore para controlar el cruce vial de la Figura 14.

Figura 14. Cruce vial



Veamos las condiciones:

- Cruce de dos vías, cada una con un solo sentido.
- Si no detecta carros en ninguna de las dos vías, repite el ciclo temporizado.
- Si detecta carros en ambas vías, repite el ciclo temporizado.
- Si detecta un vehículo en la vía 1, y no detecta en la vía 2, alarga el paso sobre la vía 1 dando otro ciclo en verde.
- Si detecta un vehículo en la vía 2, y no detecta en la vía 1, alarga el paso sobre la vía 2 dando otro ciclo en verde.

Antes que un semáforo pase a rojo, debe permanecer unos segundos en amarillo. Lista de estados: Se tiene un sistema con 4 estados. Un tiempo t1 para el tiempo que debe permanecer en verde y un t2 para el tiempo que debe permanecer en amarillo.

Tabla 3. Sistema con 4 estados

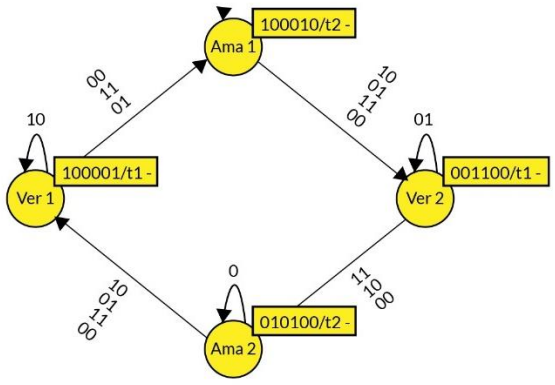
ESTADOS	**DESCRIPCIÓN**	**R2 A2 V2 R1 A1 V1 / t**
Ver 1	Semáforo 1 en verde y semáforo 2 en rojo. Las demás luces apagadas.	100001 / t1

Ama 1	Semáforo 1 en amarillo y semáforo 2 en rojo. Las demás luces apagadas.	100010 / t2
Ver 2	Semáforo 1 en rojo y semáforo 2 en verde. Las demás luces apagadas.	001100 / t1
Ama 2	Semáforo 2 en amarillo y semáforo 1 en rojo. Las demás luces apagadas.	010100 / t2

La máquina de Moore se muestra en la Figura 15:

Tabla 4. Explicación Máquina de Moore

Figura 15. Máquina de Moore



La primera secuencia indica un alto flujo vehicular sobre las dos vías. Ambos sensores están activados con 11 en todos los estados. Ver Figura 16.

La segunda secuencia detecta vehículos sobre la vía 1. Permanece 4 veces (t1) en el estado Ver1 y luego sigue el ciclo temporizado porque los sensores no detectan más vehículos.

En la tercera cadena con el primer 00 pasa a Ama1, con el segundo 00 pasa Ver2, con el siguiente 01, repite Ver2 y luego sigue el ciclo temporizado porque hay vehículos en ambas vías.

Figura 16. Sensores activados con 11 en todos los estados

Input	Result
11111111	100001/t1 - 100010/t2 - 001100/t1 - 010100/t2 - 100001/t1 -
10101000000000	100001/t1 - 100001/t1 - 100001/t1 - 100001/t1 - 100010/t2 - 001100/t1 - 010100/t2 - 100001/t1 -
000001011111	100001/t1 - 100010/t2 - 001100/t1 - 001100/t1 - 001100/t1 - 010100/t2 - 100001/t1 -

Las transiciones son las señales que envían los sensores. Cuando inicia un estado, éste permanece durante el tiempo estipulado (t1 o t2). Una vez termine el tiempo, el cambio de estado depende de los sensores.
Se observa que:

- Un ciclo en amarillo nunca se puede repetir. Una vez cumplido el tiempo t2, pasa al siguiente estado. Por eso la transición después de un estado amarillo, incluye todas las combinaciones posibles de sensores.

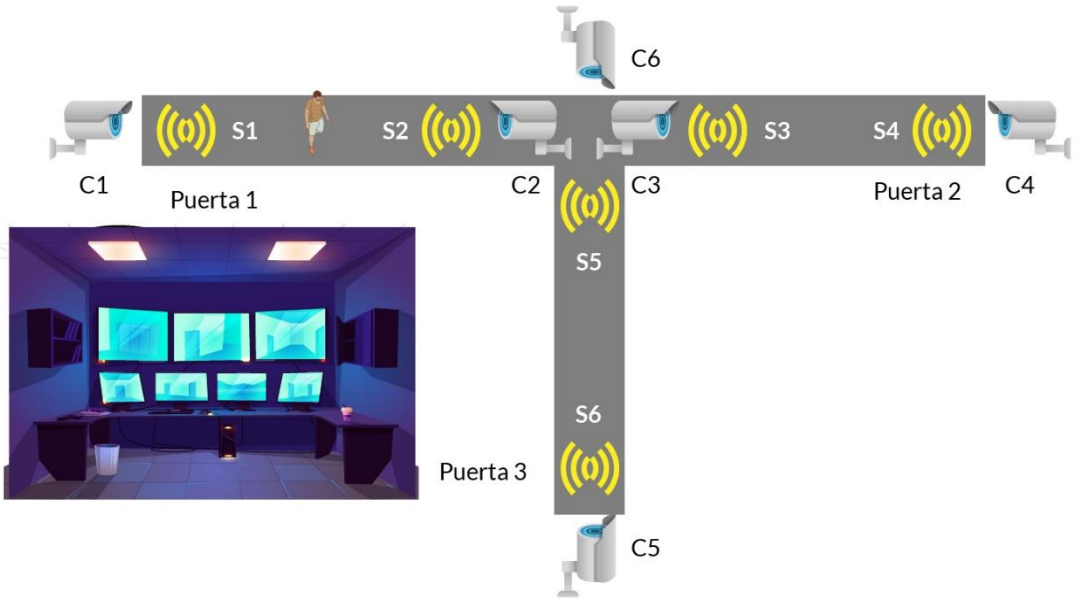
- El estado Ver1 siempre permanece un tiempo t1 y se repite solamente si detecta carros en la vía 1 y no hay carros en la vía 2 (Sensores en 10). De lo contrario pasa al estado Ama 1 donde permanecerá un tiempo t2.
- El estado Ver2 siempre permanece un tiempo t1 y se repite solamente si detecta carros en la vía 2 y no hay carros en la vía 1 (Sensores en 01). De lo contrario pasa al estado Ama 2 donde permanecerá un tiempo t2.

1.6. Máquinas secuenciales en sistemas de seguridad:

Entre las muchas aplicaciones de las máquinas secuenciales, los sistemas de vigilancia pueden ser más eficientes como puedes ver en el siguiente ejemplo.

Corredor con seis cámaras. Diseña una máquina de Mealy o de Moore que controle un circuito cerrado de tv que optimiza tiempo de grabación, grabando únicamente cuando detecta movimiento y la persona se ve de frente. Ver Figura 17.

Figura 17. Corredor con seis cámaras



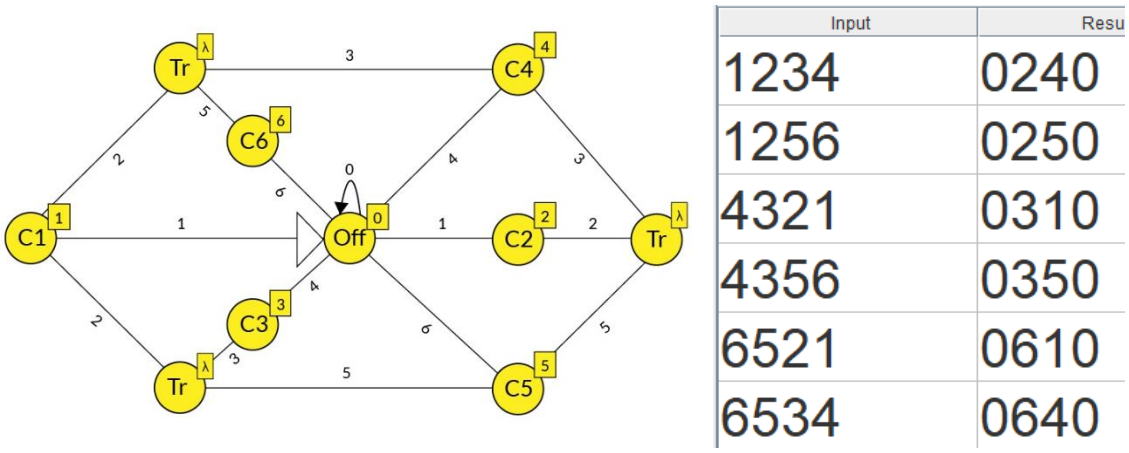
- Solamente puede haber una persona en los corredores.
- La persona debe verse de frente.
- Solamente hay un monitor.
- Se puede ver una sola cámara a la vez.
- La persona puede entrar o salir por las 3 puertas.
- Si la secuencia inicia activando S1, activa C2.
- Si la secuencia es S2-S3. activa C4.

- Si la secuencia es S2-S5 activa C5.

A continuación, te presentamos dos soluciones:

Solución 1: En la Figura 18, la máquina funciona bien, si el individuo atraviesa completamente. No reconoce la cadena si el individuo se devuelve antes de llegar hasta otra puerta. Los tres nodos "Tr", son nodos de transición que no generan señal de salida, pero sirven para crear la secuencia que se requiere. En la primera cadena, el sensor 1 prende la cámara 2. Luego la secuencia "23" prende la cámara 4 y el sensor 4 apaga todo y vuelve al estado "Off". De la misma forma, la cámara 1 se prende con la secuencia "52" o con "32" y la cámara 5 se prende con la secuencia "35" o con "25".

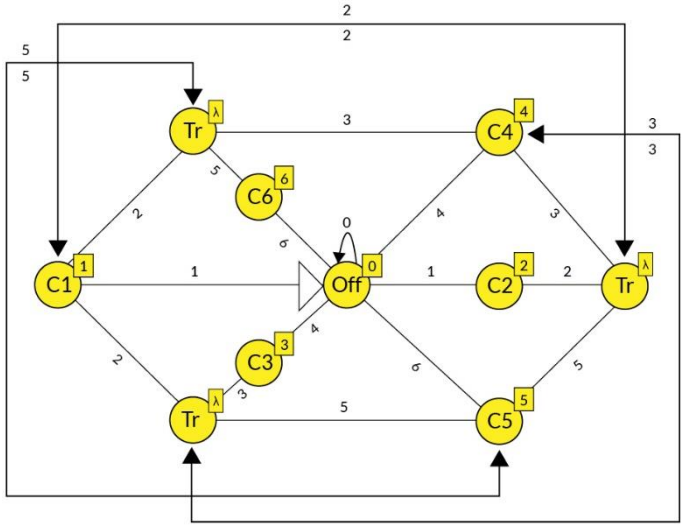
Figura 18. Buen funcionamiento de la máquina



Input	Resu
1234	0240
1256	0250
4321	0310
4356	0350
6521	0610
6534	0640

Solución 2: La máquina de la Figura 19 permite devoluciones y puede seguir al individuo con las cámaras hasta que sale del laberinto. La última cadena lee un recorrido con varias devoluciones.

Figura 19. Máquina permite devoluciones



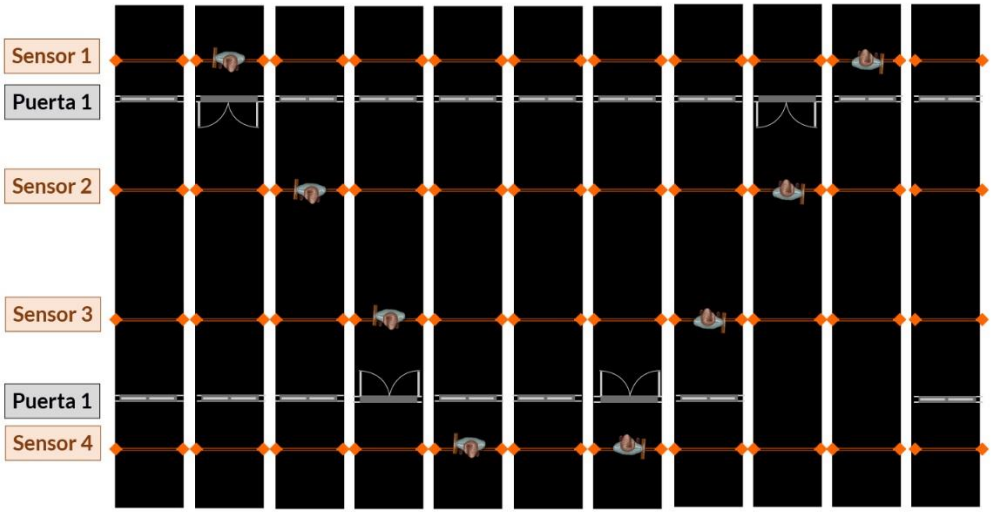
Input	Re
1234	0240
1256	0250
4321	0310
4356	0350
6521	0610
6534	0640
11	020
1221	0210
123321	02410
125521	02510
44	030
4334	0340
432234	03140
435534	03540
66	060
6556	0650
652256	06150
653356	06450
12552234	025140

1.7 Otra aplicación en sistemas de seguridad:

En este caso, la máquina secuencial no activa cámaras, sino compuertas que permiten el paso de una persona a zonas restringidas.

Exclusa de seguridad.
Diseña una máquina de Moore para controlar las puertas A y B dependiendo de si un individuo está entrando o saliendo. Genera mensaje de error si la secuencia es incorrecta.
La secuencia de detección con los momentos de apertura y cierre se aprecia la siguiente figura.

Figura 20. Secuencia de detección



- Solamente puede transitar una sola persona.
- Por defecto, ambas puertas están cerradas.
- Puede caminar de A hacia B o de B hacia A.
- Las puertas nunca pueden estar abiertas simultáneamente.
- Nunca hay dos o más sensores activados simultáneamente.
- Debe estar en estado “libre” antes de iniciar un nuevo ciclo.

Lenguaje sensores Σ = 1, 2, 3, 4.

Los estados son:

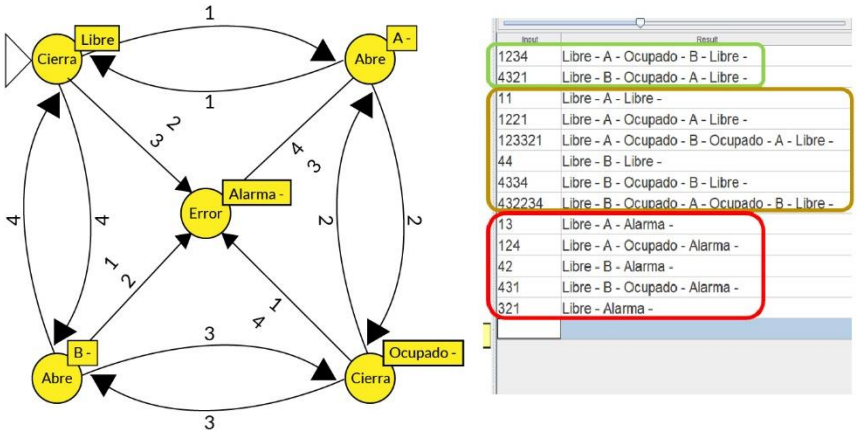
- Cierra libre (Inicial) La exclusa está libre y las dos puertas cerradas.
- Abre A: Se abre solamente la puerta “A”.
- Cierra Ocupado: Las dos puertas están cerradas y el individuo está adentro.
- Abre B: Se abre solamente la puerta “B”.
- Error: Cuando ocurre una secuencia ilógica, como pasar S1 a S3 sin activar antes el S2.

Las dos primeras cadenas, de la ilustración 22, indican que el individuo entró y pasó hasta el otro lado, inicia en “libre” y termina en “libre”.

Las seis siguientes muestran recorridos con devoluciones, inicia en “libre” y termina en “libre”.

Las cinco últimas, muestran cadenas imposibles de aceptar porque estarían saltándose al menos un sensor, inicia en “libre” y termina en “Alarma”. Que puede ser un timbre o una ventana emergente en la pantalla de un monitor.

Figura 21. Estados de la exclusa



2. Máquinas de Turing

2.1. Historia de las máquinas de Turing

En la década de los 30, el inglés Allan Turing diseñó el modelo matemático de una máquina teórica con un gran poder computacional, llamada Máquina de Turing (M.T.). Analizaremos los motivos científicos históricos que llevaron a Turing a diseñar esta máquina. (Málaga, 2008).

Russell y Whitehead desarrollaron un sistema matemático de axiomas y reglas de inferencia altamente formalizada, cuyo propósito era poder traducir a este esquema cualquier razonamiento matemático correcto. Las reglas estaban cuidadosamente seleccionadas, para evitar planteamientos que pudieran llevar a conclusiones paradójicas.

Simultáneamente, el prestigioso matemático alemán **David Hilbert** comenzó la tarea de establecer un esquema mucho más completo y manejable. Hilbert pretendía demostrar que el sistema estaba libre de contradicciones. Nunca vaciló en proclamar su convicción que algún día **con este sistema se podría resolver cualquier problema matemático o demostrar que carecía de solución**. Aseguraba que existía un procedimiento por medio del cual era posible afirmar, a priori, si un problema podía o no ser resuelto. Llamó a este problema Entscheidungs problema ("el problema de la decisión").

Sin embargo, un joven austríaco, llamado **Kurt Godel,** publicó en 1931 un artículo titulado "Sobre proposiciones formalmente indecidibles de los fundamentos de las matemáticas y sistemas relacionales". En él, Godel probó el Teorema de la "incompletitud", en el que se afirman dos importantes cuestiones:

- *Si la teoría axiomática de conjuntos es consistente, existen teoremas que no pueden ser probados ni refutados.*

- *No existe ningún procedimiento constructivo que pruebe que la teoría axiomática de conjuntos es consistente.*

En la demostración de este teorema, Godel también propuso un interesante método para enumerar objetos que, originalmente, no parecían ser enumerables. Utilizó para ello el teorema fundamental de la Aritmética o teorema de factorización única, que afirma que todo entero positivo se puede representar de forma única como producto de factores primos.

Para los científicos mencionados anteriormente, era fundamental el concepto de algoritmo o proceso efectivo, considerándolo como un método para resolver un problema genérico en un número finito de pasos, mediante operaciones conocidas y realizables. Desde este punto de vista, Turing diseñó su máquina como un dispositivo capaz de realizar un algoritmo.

Apoyó las teorías de Godel al demostrar que algunos problemas no pueden resolverse con una Máquina de Turing (M.T).

La gran ventaja de la M.T. es que, a pesar de su simplicidad, tiene un gran poder computacional y no se ve limitada por las características tecnológicas de una computadora, como la velocidad de procesamiento o la capacidad de la memoria. Por esta razón, se la considera como un símbolo invariante de la informática.

En esta misma época, Emil Post, así como Churchy Kleene, realizaron estudios similares a los de Turing. (Málaga, 2008)

Utilidad de las Máquinas de Turing (MT):

- Gracias a ser tan simples, resulta más fácil demostrar que algo no se puede resolver con ellas.
- Gracias a su equivalencia con los lenguajes de programación, facilitan entonces la demostración de que cierto problema, no se puede resolver con un lenguaje de programación.
- Mecanismo de cómputo muy sencillo de definir. Pero tan potentes como los lenguajes de programación de muy bajo nivel.
- Por ser tan demasiado de "Bajo Nivel", no resultan prácticas para programar.

2.2 Funcionamiento de la máquina de Turing

La Máquina de Turing es un mecanismo de computación notoriamente primitivo y, sin embargo, permite llevar a cabo cualquier cómputo que podamos hacer en nuestro PC (Navarro, 2016).

En la unidad 1, aprendiste que los autómatas de pila tienen unidad de memoria, pero solamente se pueden almacenar o extraer datos desde el tope de la pila, lo cual, limita el funcionamiento para muchos casos. La máquina de Turing, en lugar de la pila, tiene una cinta a la cual se le pueden extraer e intercambiar datos en cualquier parte, incluyendo los dos extremos.

Existen varios tipos de máquinas de Turing. La que vas a estudiar en esta unidad, es una máquina de una sola cinta que no está acotada en ninguna de las dos direcciones.
También existen las máquinas de Turing de varias cintas, de varios cabezales, de una sola dirección, acotadas en los extremos, pero no son de interés en este curso.
La máquina de Turing se define con una séptupla:

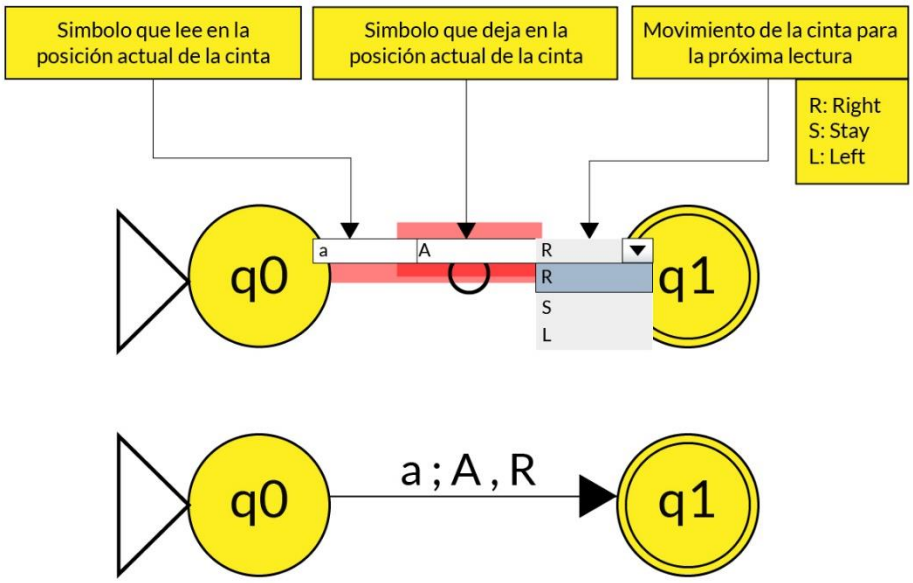
Tabla 5. Séptula

Q = es el conjunto de estados. **Σ** = es el alfabeto de entrada.	**F ⊆ Q** es el estado final.

Γ = es el alfabeto de la cinta. **s ∈ Q** es el estado inicial.	□ **⊆ Γ** es el símbolo blanco. Es el único símbolo que se puede repetir un número infinito de veces δ= función de transición, donde L es un movimiento de la cinta a la izquierda y R es el movimiento de la cinta a la derecha.

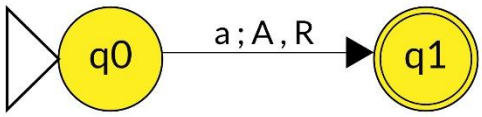
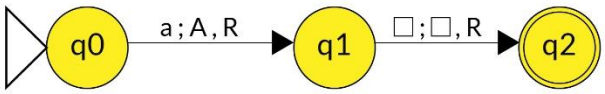
Las transiciones en la máquina de Turing tienen tres campos como se observa en la Figura 22.

Figura 22. Transiciones en la máquina de Turing



La importancia del símbolo blanco "□": La máquina de Turing se detiene cuando llegue al estado final, pero puede no haber leído toda la cadena. Observe estos dos ejemplos en la siguiente tabla.

Tabla 6. La importancia del símbolo blanco

<table>
<tr>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td>

Input	Result
a	Accept
aa	Accept
ab	Accept
b	Reject
ba	Reject

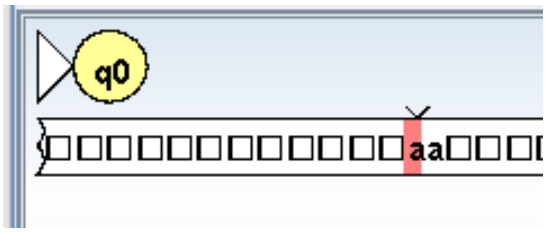
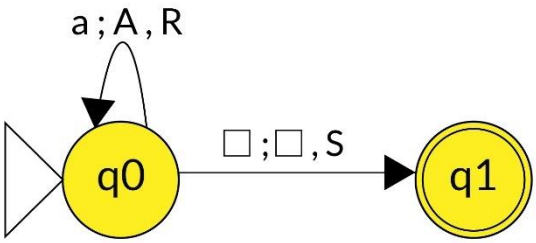
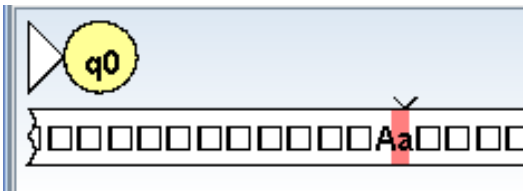
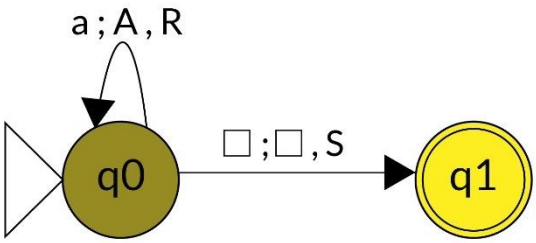
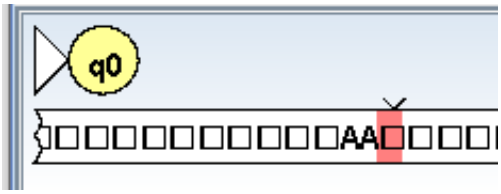
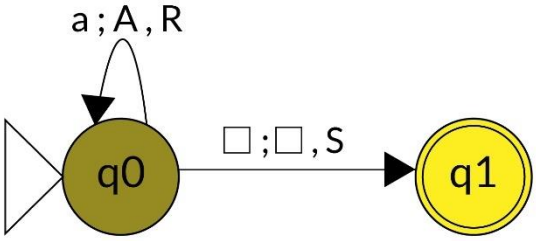
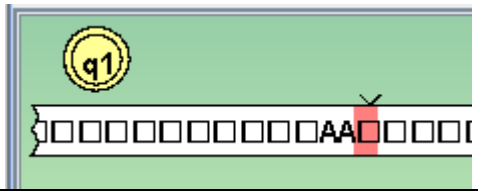
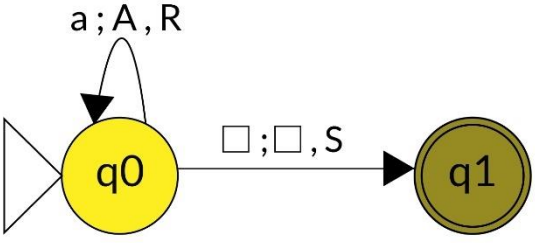
</td>
<td>

Input	Result
a	Accept
aa	Reject
ab	Reject
b	Reject
ba	Reject

</td>
</tr>
<tr>
<td>Reconoce las cadenas que inician con “a”. Se detiene sin terminar de leer y acepta la cadena como se muestra a continuación.
 
</td>
<td>Reconoce cadenas que tengan solamente una “a”. Se detiene cuando lee “□” y acepta la cadena, como se presenta a continuación.
 
</td>
</tr>
</table>

Para entender el concepto de cinta y de función de transición, observa el siguiente ejemplo:

Tabla 7. Convertir una cadena de “a” minúsculas en “A” mayúsculas. Si la entrada es “aa” la salida será “AA”

<table>
<tr>
<td>1. Esta es la cadena inicial:
 
</td>
<td>2. Cuando lea la primera “a”, la cambiará por una “A” y se moverá a la derecha (R), como se muestra a continuación.
 
</td>
</tr>
<tr>
<td>3. Ya convirtió la primera “a” en “A” y la cinta se movió una casilla a la derecha como se presenta a continuación:
 
</td>
<td>4. El autómata sigue en q0
 
</td>
</tr>
<tr>
<td>5. Ya convirtió la segunda “a” en “A” y se movió nuevamente a la derecha.
 
</td>
<td>6. Continúa en q0.
 
</td>
</tr>
<tr>
<td>7. Como leyó el espacio en blanco, dejó el mismo espacio, pero no se movió porque la función de transición es S. El fondo cambia de color indicando que reconoció la cadena completa y llegó al estado final q1, como se muestra a continuación.
 
</td>
<td>8. Ahora pasó a q1 y llegó al estado final.
 
</td>
</tr>
</table>

A pesar de su simple funcionamiento, estos autómatas resuelven cualquier problema computable. Las computadoras modernas tienen ventajas en capacidad de memoria, en velocidad y en poder trabajar con lenguajes de alto nivel, sin embargo, no hay problema que resuelva una computadora moderna, que no se pueda resolver con una máquina de Turing.

En la primera unidad viste que un autómata finito puede reconocer una expresión gramatical como a*, a^n, $a^n b^r$, etc. Un autómata de pila puede reconocer expresiones gramaticales como $a^n b^n$, porque guarda en una memoria la cantidad de veces que ha leído "a" para poder leer "b" la misma cantidad de veces. Pero el autómata de pila no puede reconocer la expresión $a^n b^n c^n$, porque la memoria tipo pila solamente la puede usar una vez.

2.3 Entrenamiento con máquinas de Turing de una cinta.

Una máquina de Turing puede entenderse como el algoritmo de los algoritmos. Puede hacer todo lo que hacen los autómatas finitos, los autómatas de pila y mucho más. En el siguiente ejemplo verás un caso de reconocimiento de cadenas que no se podría resolver con autómatas finitos o de pila. Los autómatas finitos no tienen memoria. Los autómatas de pila tienen una memoria que se puede usar solamente una vez. La máquina de Turing, resuelve ese problema.

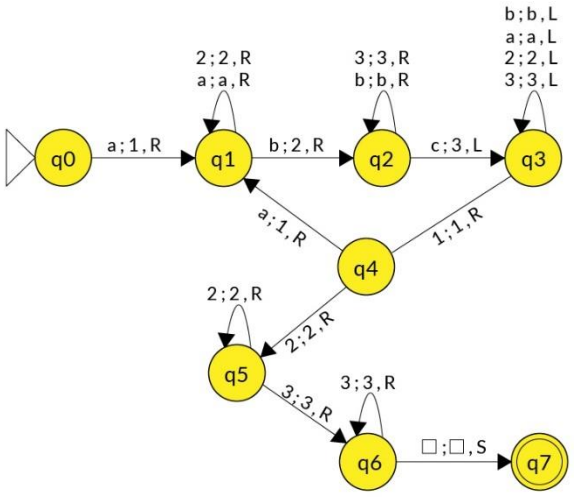
2.3.1 Reconocimiento de cadenas con memoria ilimitada:

Diseña una máquina de Turing que reconozca la expresión $a^n b^n c^n$.
El alfabeto de entrada $\Sigma = \{a, b, c\}$
El alfabeto de la cinta: $\Gamma = \{1, 2, 3\}$

Funcionamiento: La máquina solamente reconoce cadenas que inician con "a", luego continúan con "b" y finalmente con "c". Si encuentra una "b" antes que una "a", o una "c" antes que una "b", debe rechazarla. Además, debe asegurar que la cantidad de veces que lea cada letra del alfabeto Σ, sea siempre igual a "n". Por ejemplo, si n=2, la cadena es $a^2 b^2 c^2$, o sea, "aabbcc". Si n=4, la cadena es $a^4 b^4 c^4$, o sea, "aaaabbbbcccc".

El algoritmo de la siguiente figura se basa en hacer un primer recorrido hacia la derecha y reemplazar la primera "a" por un "1", (q0 →q1) la primera "b" por un "2" (q1 →q2) y la primera "c" por un "3" (q2 →q3). Las demás letras que lee las deja tal cual.

Figura 23. Algoritmo de reemplazo “a”, “b” y “c”



Input	Result
abc	Accept
aabbcc	Accept
aaabbbccc	Accept
aaaaabbbbbccccc	Accept
aaabbcc	Reject
aabbccc	Reject
bbaacc	Reject
aaabbbcc	Reject
ccbbaa	Reject
aaccbb	Reject

En q3, la cinta se empieza a mover a la izquierda, hasta que lee el “1” que reemplazó la primera “a” (q3 →q4). Entonces la cinta cambia de sentido, nuevamente a la derecha y si lee otra “a”, la reemplaza por “1”, la segunda “b” por “2” y la segunda “c” por “3”. Se repite este ciclo hasta que ya no encuentra ninguna “a” y no puede volver a q1.
Como las tres letras se repiten la misma cantidad de veces, cuando haya reemplazado todas las “c” por “3”, también debe haber reemplazado todas las “b” por “2” y todas las “a” por “1”. Por lo tanto, en el recorrido final, a partir de q4, solamente debe leer los “2” y luego los “3” que quedaron en la cadena.

Cadenas que rechaza:

Si la letra que menos se repite es la “a”, no podrá llegar al final, porque tratará de dirigirse a q5 donde solamente lee acepta “2” o a q6 donde solamente acepta “3”. Como aún hay alguna “b” o “c” en la cadena, en alguno de estos dos nodos la rechaza.
Si la letra que menos se repite es la “b”, habrá un ciclo que llegue hasta q1, pero no podrá leer la transición (q1 →q2).
Si la letra que menos se repite es la “c”, habrá un ciclo que llegue hasta q2, pero no podrá leer la transición (q2 →q3).

2.3.2. Algoritmo de ordenamiento con dos letras:

Dado el alfabeto Σ= {a, b}, diseña una máquina de Turing que ordene alfabéticamente la cadena leída. Por ejemplo, si lee “abbaba” debe generar “aaabbb”.

La máquina de Turing de la siguiente figura hace este ordenamiento. Funciona de la siguiente forma:

Figura 24. Algoritmo ordenamiento con dos letras

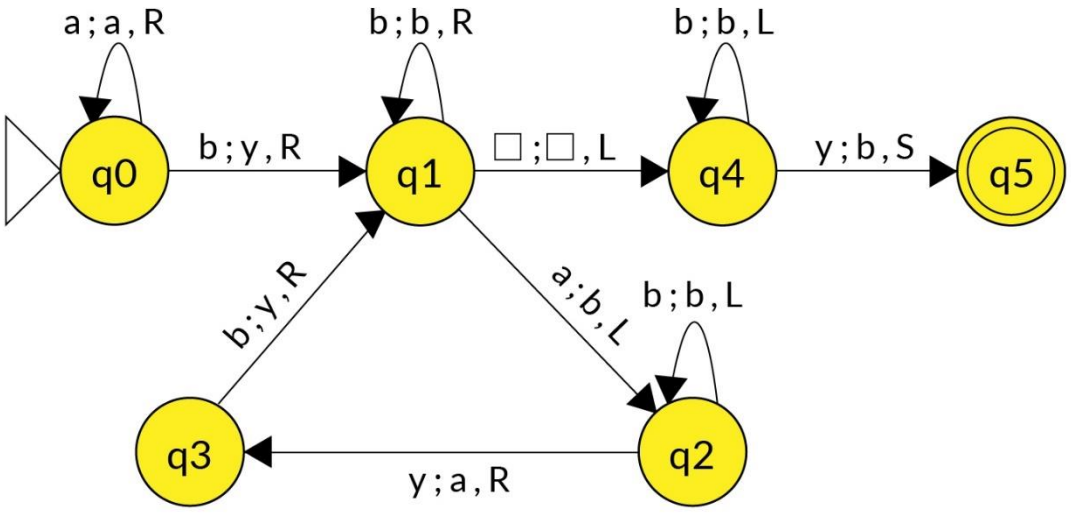
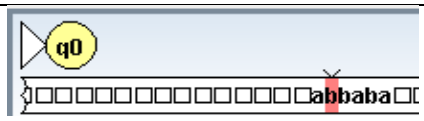
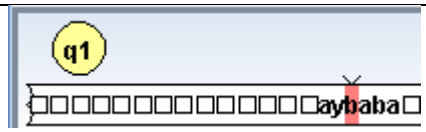
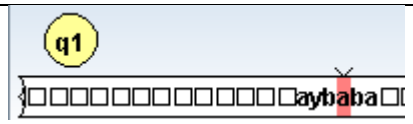
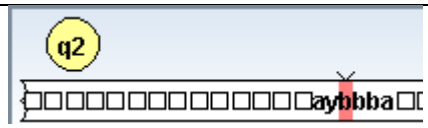
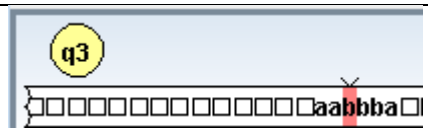
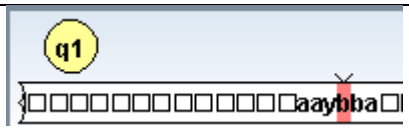
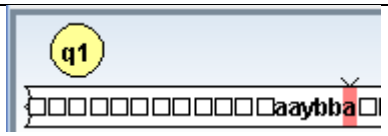
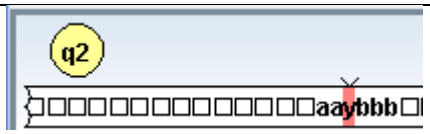
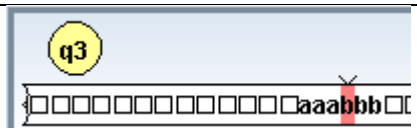
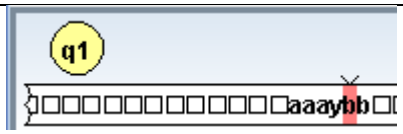
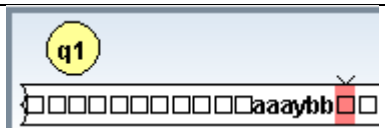
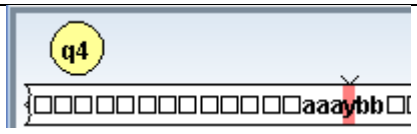
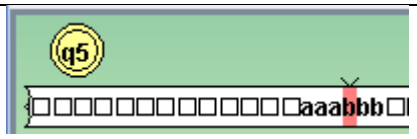


Tabla 8. Pasos del ordenamiento

1. En (q0), las "a" que hay al inicio la dejas donde está y se mueve a la derecha hasta encontrar la primera "b".	
2. En (q0→q1) tomas la primera "b" prestada de la cadena y dejas como depósito una "*y*" para marcar la ubicación donde termina la primera tanda de letras "a".	
3. En q1, encuentras varias "b", las dejas como las encuentra moviéndote hacia la derecha hasta (q1→q2), donde encuentras la siguiente "a".	
4. Allí, debes tomar prestada la "a" y devuelves la "b" que habías tomado prestada en el paso anterior. Ahora le debes una "a" a la cadena, pero sabes que a la izquierda hay una "*y*" que dejaste marcando la posición.	
5. En q2 recorres hacia la izquierda todas las "b" que habías leído en q1, hasta encontrar la "*y*". En (q2 →q3), reemplazas la "*y*" por la "a" que habías tomado prestada en el paso anterior y te mueves una casilla a la derecha.	

Hasta este punto, la cadena debe estar así: "a**a**b**b**ba". Intercambiaste una "a" por la primera "b" que había en la cadena. A la izquierda de la posición actual, solamente hay letras "a".

Tabla 9. Continuación pasos del ordenamiento

6. Ahora mueves la cinta nuevamente, una posición a la derecha, en (q3 →q1), tomas prestada la "b" que encuentras y dejas como depósito otra vez la "y".	
7. Sigues a la derecha hasta que vuelves a encontrar otra "a"	
8. Reemplazas la "a" por la "b" que habías tomado prestada y arrancas a la izquierda en busca de la "*y*",	
9. Reemplaza la "*y*" por la "a" que debía y se desplaza una casilla ala derecha. Aunque la cadena está ordenada, el autómata no sabe que se acabaron las "a".	
10. Para terminar el ciclo, quita una "b" y deja como depósito una "*y*". Observa que la "*y*" siempre queda donde termina la primera tanda de "a" y empieza la primera tanda de "b".	
11. Como ya no hay más "a", en este ciclo, se termina la cadena y encuentra un blanco "□". En (q1 →q4), la cinta vuelve a cambiar de sentido. Recuerda que hay una "*y*" que quedó marcando la posición donde terminan las "a" y comienzan las "b".	
12. Por lo tanto, hay que moverse a la izquierda en q4 sin alterar la cadena, hasta llegar a la "*y*".	
13. En (q4 →q5), cambias la "*y*" por la "b" que le debías a la cadena. En ese momento puedes finalizar.	

2.3.3. Algoritmo que calcula el consecutivo de un número dado en binario:

A continuación, hay un pantallazo en la ilustración 54 con algunas tandas de números binarios consecutivos que te pueden servir para comprobar el autómata.

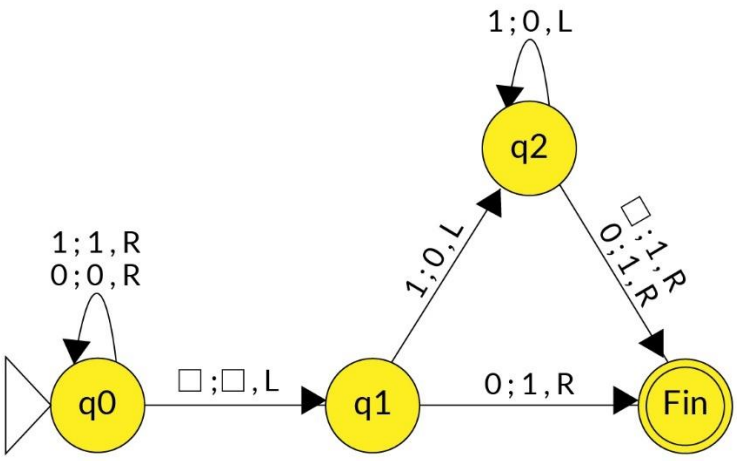
Figura 25. Números binarios consecutivos para comprobar el autómata

Dec	Bin		Dec	Bin
0	0		30	11110
1	1		31	11111
2	10		32	100000
3	11		33	100001
4	100		50	110010
5	101		51	110011
6	110		52	110100
7	111		53	110101
8	1000		70	1000110
9	1001		71	1000111
10	1010		72	1001000
11	1011		73	1001001
12	1100		90	1011010
13	1101		91	1011011
14	1110		92	1011100
15	1111		100	1100100
16	10000		101	1100101

En la siguiente figura se aprecia que la máquina inicia recorriendo la cadena hasta el extremo derecho, cuando detecta el blanco "□", la cinta cambia de sentido. Si la última cifra es 0, se cambia por un 1 y termina.

Si el número termina en 1, se cambia la última cifra por un 0 y todo los unos a la izquierda se reemplazan por ceros, hasta encontrar un cero o un blanco, el cual se reemplaza por un 1 y así llega al estado final.

Figura 26. Inicio de recorrido de la cadena hasta el extremo derecho



Monta la máquina virtual en tu computador y comprueba el funcionamiento. Si tienes otra solución, la compartes con tu profesor y tus compañeros en el encuentro sincrónico

2.4 Máquinas de Turing en sistema unario.

Las matemáticas pueden operar con muchos sistemas numéricos. Los más conocidos son: el sistema binario con dos símbolos, el 0 y el 1. El sistema decimal con 10 símbolos que son los dígitos del 0 al 9 y el sistema hexadecimal, con 16 símbolos que son los dígitos del 0 al 9 más las letras de la "A" a la "F".

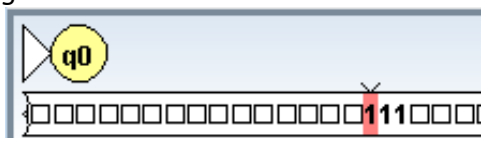
El sistema unario tiene un solo símbolo, el 1. Y sin darnos cuenta lo hemos estado usando toda la vida, como se ve a continuación:

Figura 27. Sistema unario

A continuación, verás un ejemplo con una operación matemática en sistema unario.

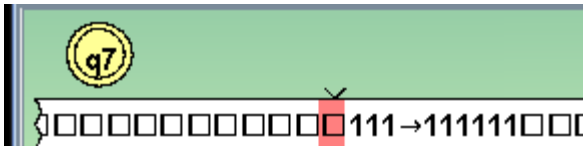
Máquina de Turing que multiplica por dos en sistema unario. Existen varios algoritmos para esta operación. A continuación, veremos:

Figura 28. Resultado de la cadena inicial



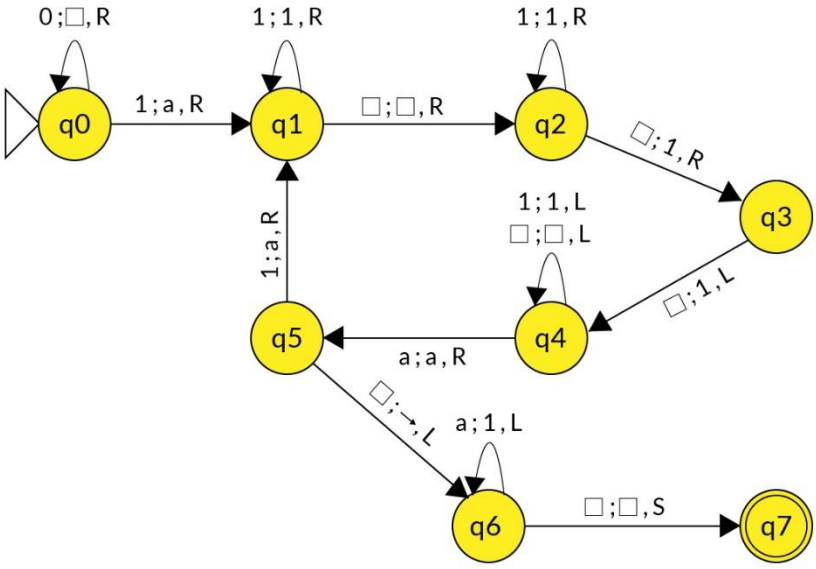
Pudiste ver uno que da como resultado la cadena inicial entre los signos < y > y finalmente el resultado de la multiplicación en la siguiente figura. Adiciona un "11" en la cadena de la derecha, por cada "1" que lee en la cadena de la izquierda.

Figura 29. Resultado de la multiplicación



El algoritmo de figura 30, inicia reemplazando el primer “1” por una “a” (q0→q1), para llevar la cuenta de cuantos “1” ha leído. Al final de la cadena inicial de unos, deja un espacio blanco “□” (q1→q2), para escribir el resultado al lado derecho. Pasa al lado derecho de la ecuación y genera el “11” en (q2→q3→q4). Cambia el sentido de la cinta a la izquierda hasta que lee la “a” y vuelve a cambiar el movimiento a la derecha repitiendo el ciclo si lee un “1” hasta terminar la cadena inicial. En el último ciclo, se mueve a la izquierda hasta detectar la “a” en (q4→q5), luego a la derecha y como lee el espacio blanco “□” en lugar del “1” en (q5→q6), significa que ya no hay más “1”. Deja el signo “→” y se mueve a la izquierda reemplazando todas las “a” por unos para mostrar la cadena inicial.

Figura 30. Algoritmo cuantos “1” ha leído



La simulación la vemos a continuación.

Figura 31. Simulación

Input	Output
1	<1>11
11	<11>1111
111	<111>111111
1111	<1111>11111111
11111	<11111>1111111111

2.5 Entrenamiento con máquinas de Turing multi-cinta

Las máquinas de Turing multi-cinta, son útiles cuando los algoritmos son muy largos. Por ejemplo, reconocer una cadena como $a^nb^nc^nd^n$ es posible en una máquina de una sola cinta, pero requiere de muchos estados y muchas transiciones. Igualmente ordenar una cadena con un alfabeto Σ= {a, b, c, d} en una máquina de una sola cinta, sería un grafo muy complejo. En una máquina multi-cinta, estos algoritmos son mucho más simples.

Cierre

Las máquinas secuenciales, vistas al inicio de la unidad, permiten dar un salto de la teoría a la práctica. De cierta forma responden la pregunta que debe hacerse todo tecnólogo. Y ¿para qué sirven los autómatas? La segunda unidad de esta asignatura está mostrando aplicaciones de los autómatas en el mundo real. Desde aplicaciones blandas, como reconocimientos de contraseñas, hasta aplicaciones duras como automatización de máquinas.

Las máquinas de Turing, vistas en la última sección, pueden resolver cualquier algoritmo lógico o matemático. En los últimos años se han visto algunos avances que permiten aumentar la velocidad de procesamiento, la seguridad, la facilidad de programación, etc. Sin embargo, sigue siendo el fundamento básico de la computación moderna. Para cerrar este capítulo, puedes entender una máquina de Turing como EL ALGORITMO PARA HACER ALGORITMOS.

Bibliografía

- *Amaya, J. M. (2015). Autómatas y Lenguajes Formales. Duitama, Boyacá, Colombia.*
- *Cueva Lovelle, J.M; Izquierdo Castanedo, R; Luengo Díez, M. C; Ortín Soler, F; Labra Gayo, J.E. (2003) Lenguajes, Gramáticas y Autómatas en Procesadores de Lenguaje. Ed. SERVITEC. Universidad de Oviedo.*
- *Acevedo Juárez, A. (2006) Teoría de la computación. Instituto Tecnológico de Celaya. Tesis de grado.*
- *Jurado Málaga, E (2008) Teoría de autómatas y lenguajes formales. Espacio Europeo Educación Superior (E.E.E.S.) Universidad de Extremadura.*
- *Chacón, J. L. (2005). Lenguajes y Autómatas finitos. Recuperado de:* https://openlibra.com/es/book/download/lenguajes-y-automatas-finitos
- *Huertas, M., (2011). Teoría de conjuntos básica. Recuperado de:* https://openlibra.com/es/book/teoria-de-conjuntos-basica
- *Hopcroft, J., & Ullman, J. (1969). Their Formal Languages Relation to Automata. Recuperado de:* https://openlibra.com/es/book/formal-languages-and-their-relation-to-automata
- *Hurtado, J. A., & Kantor, R., & Luna, C., & Sierra, L., & Zanarini, D. (2014). Temas de Teoría de la Computación . Recuperado de:* https://openlibra.com/es/book/download/temas-de-teoria-de-la-computacion
- *Ivorra, C. (2010). Teoría Descriptiva de Conjuntos I. Recuperado de:* https://www.uv.es/Ivorra/Libros/Descriptiva2I.pdf
- *Ivorra C. (2011). Lógica y Teoría de Conjuntos. Recuperado de:* https://openlibra.com/es/book/download/logica-y-teoria-de-conjuntos.
- *Navarro, G. (2017). Hacia Teoría de la Computación: Lenguajes Formales, Computabilidad y Complejidad . Recuperado de:* https://openlibra.com/es/book/teoria-de-la-computacion-lenguajes-formales-computabilidad-y-complejidad

- *Ortega, J. L. (2004). Breves Notas sobre Autómatas y Lenguajes. Recuperado de:* https://openlibra.com/es/book/download/breves-notas-sobre-automatas-y-lenguajes

- *Proyecto Latín. (2016). Temas de Teoría de la Computación. Tomo 1. Recuperado de:* https://openlibra.com/es/book/temas-de-teoria-de-la-computacion

IU Digital de Antioquia

INSTITUCIÓN UNIVERSITARIA DIGITAL DE ANTIOQUIA