

**INSTITUTO TECNOLÓGICO DE IZTAPALAPA**

ING. SISTEMAS COMPUTACIONALES

**INFORME TÉCNICO DE RESIDENCIA**

**PROFESIONAL**

**TEMA:**

PUERTAS LOGICAS Y TEORIA DE CIRCUITOS

**LUGAR DE REALIZACIÓN:**

INSTITUTO TECNOLOGICO DE IZTAPALAPA

**ASESOR INTERNO:**

ING PARRA HERNANDEZ ABIEL TOMAS

**ASESOR EXTERNO**:

-

**PRESENTA:**

RAMIREZ MARTINEZ IVAN VIDAL

**No DE CONTROL:**

171080137

**CIUDAD DE MÉXICO MAY / 2021**

Resumen.

En este informe se realizó la investigación de los métodos computacionales, los cuales fueron puertas lógicas y teoría de circuitos para la materia de Lenguajes y Autómatas l, el objetivo principal para este informe es conocer y entender  que es un [modelo de cálculo](https://en.wikipedia.org/wiki/Model_of_computation) en el que los valores de entrada proceden a través de una secuencia de puertas, cada una de las cuales calcula una [función](https://en.wikipedia.org/wiki/Function_(computer_science)) . Los circuitos de este tipo proporcionan una generalización de [los circuitos booleanos](https://en.wikipedia.org/wiki/Boolean_circuit) y un modelo matemático para circuitos [lógicos](https://en.wikipedia.org/wiki/Digital_logic) digitales. Los circuitos se definen por las puertas que contienen y los valores que pueden producir las puertas y comparar los modelos de computación con la máquina de Turing.

Índice.

Introduccion…………………………………………………………………4

Justificacion…………………………………………………………………5

Objetivos…………………………………………………………………….6

Capitulo 1. Marco teorico.

* 1. Puerta lógica………………………………………………………….7
  2. Teoría de circuitos……………………………………………………10
  3. Jerarquía de Chomsky……………………………………………….11
  4. Teoría de autómatas………………………………………………….14

Capitulo 2. Metodología.

2.1. Puerta lógica, ejercicio 1……….………………………………………15

2.2. Puerta lógica, ejercicio 2……………………………………………….16

2.3. Teoria de circuitos ejercicio 1………………………………………….17

Conclusiones ……………………………………………………………….29

Referencias bibliográficas y virtuales. ……………………………………………30

Índice de figuras.

Figura 2.1. Tabla de familia lógica…………………………………………………………………8

Figura 2.3 jerarquía de Chomsky……………………………………………………………………11

Figura 2.4 Teoría de autómatas……………………………………………………………………14

Introducción.

En este reporte se hablara sobre las puertas lógicas y teoría de circuitos donde conoceremos los  [modelo](https://en.wikipedia.org/wiki/Model_of_computation) idealizado [de computación](https://en.wikipedia.org/wiki/Model_of_computation) o dispositivo [electrónico](https://en.wikipedia.org/wiki/Electronics) físico que implementa una [función booleana](https://en.wikipedia.org/wiki/Boolean_function) , una [operación lógica](https://en.wikipedia.org/wiki/Logical_operation) realizada en una o más entradas [binarias](https://en.wikipedia.org/wiki/Binary_number) que produce una única salida binaria. Dependiendo del contexto, el término puede referirse a una puerta lógica ideal , una que tiene, por ejemplo, un [tiempo de subida](https://en.wikipedia.org/wiki/Rise_time) cero y un [abanico](https://en.wikipedia.org/wiki/Fan-out) ilimitado , o puede referirse a un dispositivo físico no ideal. Donde también incluyen dispositivos tales como [multiplexores](https://en.wikipedia.org/wiki/Multiplexer) , [registros](https://en.wikipedia.org/wiki/Processor_register) , [unidades lógicas aritméticas](https://en.wikipedia.org/wiki/Arithmetic_logic_unit) (ALU) y [memoria de computadora](https://en.wikipedia.org/wiki/Computer_memory) , hasta [microprocesadores](https://en.wikipedia.org/wiki/Microprocessor) completos , que pueden contener más de 100 millones de puertas. En la práctica moderna, la mayoría de las puertas están hechas de [MOSFET](https://en.wikipedia.org/wiki/MOSFET) ( [transistores de efecto de campo](https://en.wikipedia.org/wiki/Field-effect_transistor) semiconductores de óxido metálico ).

Justificación.

La carrera de ingeniería en sistemas computacionales que imparte el tecnológico de Iztapalapa tiene como objetivo diseñar, configurar y administrar redes computacionales aplicando las normas y estándares vigentes, administración de base de datos, sabrá implementar las tecnologías y tendrá las habilidades, para aplicar y resolver problemas de acuerdo a las necesidades de la empresa u organización, por lo tanto el ingeniero en sistemas computacionales debe contribuir a la innovación de tecnologías y herramientas operativas. Es por ello que el presente informe surge por la necesidad de fomentar el interés en las computadoras utilizando las herramientas y los recursos. La computadora, es un medio de apoyo informativo que permite a las personas adquirir conocimientos sobre puertas lógicas y teoría de circuitos.

Objetivos.

Conocer y entender  que es un [modelo de cálculo](https://en.wikipedia.org/wiki/Model_of_computation) en el que los valores de entrada proceden a través de una secuencia de puertas, cada una de las cuales calcula una [función](https://en.wikipedia.org/wiki/Function_(computer_science)) . Los circuitos de este tipo proporcionan una generalización de [los circuitos booleanos](https://en.wikipedia.org/wiki/Boolean_circuit) y un modelo matemático para circuitos [lógicos](https://en.wikipedia.org/wiki/Digital_logic) digitales. Los circuitos que definen por las puertas que contienen y los valores que pueden producir las puertas.

Objetivos específicos:

* Desarrollar e implementar un problema de computación con los modelos computacionales que se nos fueron asignado.
* Comparar los modelos de computación con la máquina de Turing.
* Redactar y dar a entender como funciona los modelos computacionales.

Marco teórico.

* 1. Puerta lógica.

Una puerta lógica es un [modelo](https://en.wikipedia.org/wiki/Model_of_computation) idealizado [de computación](https://en.wikipedia.org/wiki/Model_of_computation) o dispositivo [electrónico](https://en.wikipedia.org/wiki/Electronics) físico que implementa una [función booleana](https://en.wikipedia.org/wiki/Boolean_function) , una [operación lógica](https://en.wikipedia.org/wiki/Logical_operation) realizada en una ,más entradas [binarias](https://en.wikipedia.org/wiki/Binary_number) que produce una única salida binaria. Dependiendo del contexto, el término puede referirse a una puerta lógica ideal , una que tiene, por ejemplo, un [tiempo de subida](https://en.wikipedia.org/wiki/Rise_time) cero y un [abanico](https://en.wikipedia.org/wiki/Fan-out) ilimitado , o puede referirse a un dispositivo físico no ideal , (ver [Operación ideal y real). amperios](https://en.wikipedia.org/wiki/Ideal_and_real_op-amps) para comparar).

Las puertas lógicas se implementan principalmente mediante [diodos](https://en.wikipedia.org/wiki/Diode) o [transistores que](https://en.wikipedia.org/wiki/Transistor) actúan como [interruptores electrónicos](https://en.wikipedia.org/wiki/Switch#Electronic_switches) , pero también se pueden construir usando [tubos de vacío](https://en.wikipedia.org/wiki/Vacuum_tube) , [relés](https://en.wikipedia.org/wiki/Relay) electromagnéticos ( [lógica de relé](https://en.wikipedia.org/wiki/Relay_logic) ), [lógica fluídica](https://en.wikipedia.org/wiki/Fluidic_logic) , [lógica neumática](https://en.wikipedia.org/wiki/Pneumatics#Pneumatic_logic) , [óptica](https://en.wikipedia.org/wiki/Optics) , [moléculas](https://en.wikipedia.org/wiki/Molecular_logic_gate) o incluso elementos [mecánicos](https://en.wikipedia.org/wiki/Analytical_engine) . Con la amplificación, las puertas lógicas se pueden conectar en cascada de la misma manera que se pueden componer funciones booleanas, lo que permite la construcción de un modelo físico de toda [la lógica booleana](https://en.wikipedia.org/wiki/Boolean_logic) y, por lo tanto, todos los algoritmos y las [matemáticas.](https://en.wikipedia.org/wiki/Mathematics) que se puede describir con lógica booleana.

Los circuitos lógicos incluyen dispositivos tales como [multiplexores](https://en.wikipedia.org/wiki/Multiplexer) , [registros](https://en.wikipedia.org/wiki/Processor_register) , [unidades aritméticas lógicas](https://en.wikipedia.org/wiki/Arithmetic_logic_unit) (ALU) y [memoria de computadora](https://en.wikipedia.org/wiki/Computer_memory) , hasta [microprocesadores](https://en.wikipedia.org/wiki/Microprocessor) completos , que pueden contener más de 100 millones de puertas. En la práctica moderna, la mayoría de las puertas están hechas de [MOSFET](https://en.wikipedia.org/wiki/MOSFET) ( [transistores de efecto de campo](https://en.wikipedia.org/wiki/Field-effect_transistor) semiconductores de óxido metálico ).

Las puertas lógicas compuestas [AND-OR-Invert](https://en.wikipedia.org/wiki/AND-OR-Invert) (AOI) y OR-AND-Invert (OAI) se emplean a menudo en el diseño de circuitos porque su construcción utilizando MOSFET es más simple y más eficiente que la suma de las puertas individuales.

* 1. Puertas electrónicas.

Un sistema lógico [funcionalmente completo](https://en.wikipedia.org/wiki/Functionally_complete) puede estar compuesto por [relés](https://en.wikipedia.org/wiki/Relay) , [válvulas](https://en.wikipedia.org/wiki/Thermionic_valve) (tubos de vacío) o [transistores](https://en.wikipedia.org/wiki/Transistor) . La familia más simple de puertas lógicas usa [transistores bipolares](https://en.wikipedia.org/wiki/Bipolar_transistors) y se llama [lógica resistor-transistor](https://en.wikipedia.org/wiki/Resistor%E2%80%93transistor_logic) (RTL). A diferencia de las puertas lógicas de diodos simples (que no tienen un elemento de ganancia), las puertas RTL se pueden conectar en cascada indefinidamente para producir funciones lógicas más complejas. Las puertas RTL se utilizaron en los primeros [circuitos integrados](https://en.wikipedia.org/wiki/Integrated_circuit) . Para mayor velocidad y mejor densidad, las resistencias utilizadas en RTL fueron reemplazadas por diodos, lo que resultó en [una lógica de diodo-transistor](https://en.wikipedia.org/wiki/Diode%E2%80%93transistor_logic) (DTL). [Lógica transistor-transistor](https://en.wikipedia.org/wiki/Transistor%E2%80%93transistor_logic)(TTL) luego suplantó a DTL. A medida que los circuitos integrados se volvieron más complejos, los transistores bipolares fueron reemplazados por [transistores de efecto de campo](https://en.wikipedia.org/wiki/Field-effect_transistor) más pequeños ( [MOSFET](https://en.wikipedia.org/wiki/MOSFET) ); consulte [PMOS](https://en.wikipedia.org/wiki/PMOS_logic) y [NMOS](https://en.wikipedia.org/wiki/NMOS_logic) . Para reducir aún más el consumo de energía, la mayoría de las implementaciones de chips actuales de los sistemas digitales utilizan ahora lógica [CMOS](https://en.wikipedia.org/wiki/CMOS) . CMOS utiliza dispositivos MOSFET complementarios (tanto de canal n como de canal p) para lograr una alta velocidad con baja disipación de energía.

Para la lógica a pequeña escala, los diseñadores ahora usan puertas lógicas prefabricadas de familias de dispositivos como la [serie](https://en.wikipedia.org/wiki/7400_series)[TTL](https://en.wikipedia.org/wiki/Transistor%E2%80%93transistor_logic)[7400](https://en.wikipedia.org/wiki/7400_series) de [Texas Instruments](https://en.wikipedia.org/wiki/Texas_Instruments) , la [serie](https://en.wikipedia.org/wiki/4000_series)[CMOS](https://en.wikipedia.org/wiki/CMOS)[4000](https://en.wikipedia.org/wiki/4000_series) de [RCA](https://en.wikipedia.org/wiki/RCA) y sus descendientes más recientes. Cada vez más, estas puertas lógicas de función fija están siendo reemplazadas por [dispositivos lógicos programables](https://en.wikipedia.org/wiki/Programmable_logic_device) , que permiten a los diseñadores empaquetar muchas puertas lógicas mixtas en un solo circuito integrado. La naturaleza programable en campo de [los dispositivos lógicos programables](https://en.wikipedia.org/wiki/Programmable_logic_device) como los [FPGA](https://en.wikipedia.org/wiki/Field-programmable_gate_array)ha reducido la propiedad "dura" del hardware; Ahora es posible cambiar el diseño lógico de un sistema de hardware reprogramando algunos de sus componentes, lo que permite cambiar las características o la función de una implementación de hardware de un sistema lógico. Otros tipos de puertas lógicas incluyen, pero no se limitan a:

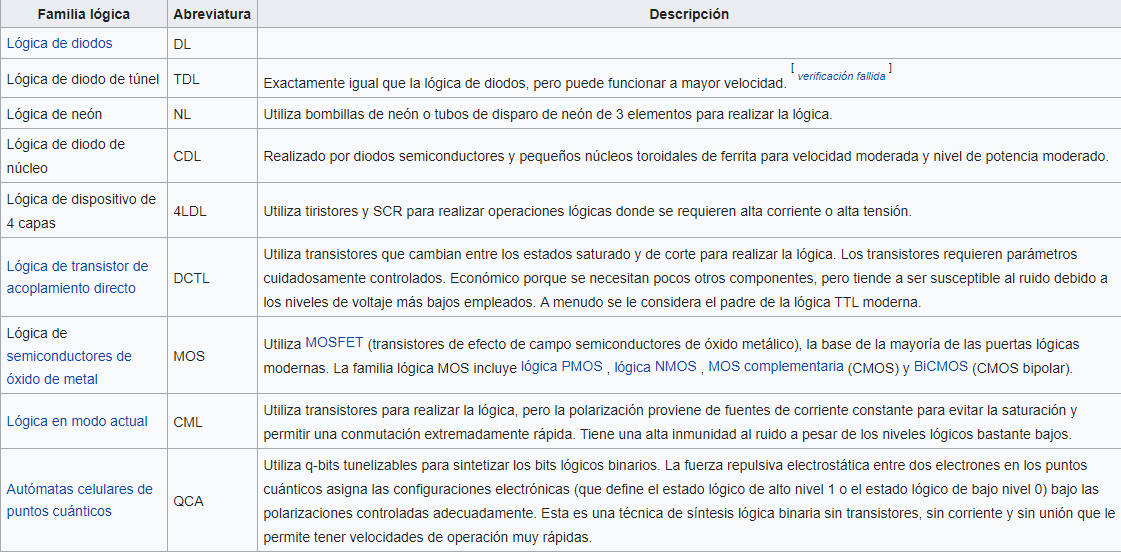


Figura 2.1. Tabla de familia lógica.

Las puertas lógicas electrónicas difieren significativamente de sus equivalentes de relés e interruptores. Son mucho más rápidos, consumen mucha menos energía y son mucho más pequeños (todo por un factor de un millón o más en la mayoría de los casos). Además, existe una diferencia estructural fundamental. El circuito de conmutación crea una ruta metálica continua para que la corriente fluya (en cualquier dirección) entre su entrada y su salida. La puerta lógica del semiconductor, por otro lado, actúa como un [amplificador de](https://en.wikipedia.org/wiki/Electronic_amplifier)[voltaje de](https://en.wikipedia.org/wiki/Voltage) alta [ganancia](https://en.wikipedia.org/wiki/Gain_(electronics)) , que absorbe una pequeña corriente en su entrada y produce un voltaje de baja impedancia en su salida. No es posible que la corriente fluya entre la salida y la entrada de una puerta lógica semiconductora.

1.2. Teoría de circuitos.

En [informática teórica](https://en.wikipedia.org/wiki/Theoretical_computer_science) , un circuito es un [modelo de cálculo](https://en.wikipedia.org/wiki/Model_of_computation) en el que los valores de entrada proceden a través de una secuencia de puertas, cada una de las cuales calcula una [función](https://en.wikipedia.org/wiki/Function_(computer_science)) . Los circuitos de este tipo proporcionan una generalización de [los circuitos booleanos](https://en.wikipedia.org/wiki/Boolean_circuit) y un modelo matemático para circuitos [lógicos](https://en.wikipedia.org/wiki/Digital_logic) digitales. Los circuitos se definen por las puertas que contienen y los valores que pueden producir las puertas. Por ejemplo, los valores en un,circuito [booleano](https://en.wikipedia.org/wiki/Boolean_data_type) son valores [booleanos](https://en.wikipedia.org/wiki/Boolean_data_type) circuito,incluye puertas,de [conjunción](https://en.wikipedia.org/wiki/Logical_conjunction) , [disyunción](https://en.wikipedia.org/wiki/Logical_disjunction) y

[negación](https://en.wikipedia.org/wiki/Logical_negation) . Los valores en un [circuito entero](https://en.wikipedia.org/wiki/Integer_circuit) son [conjuntos](https://en.wikipedia.org/wiki/Set_(mathematics)) de [enteros](https://en.wikipedia.org/wiki/Integer) y las puertas calculan la [unión de conjuntos](https://en.wikipedia.org/wiki/Set_union) , la [intersección de conjuntos](https://en.wikipedia.org/wiki/Set_intersection) y el [complemento de conjuntos](https://en.wikipedia.org/wiki/Set_complement) , así como las [operaciones aritméticas de](https://en.wikipedia.org/wiki/Arithmetic_operation)[suma](https://en.wikipedia.org/wiki/Addition) y [multiplicación](https://en.wikipedia.org/wiki/Multiplication).

1.3. Jerarquía de Chomsky.

En [la teoría del lenguaje formal](https://en.wikipedia.org/wiki/Formal_language) , [informática](https://en.wikipedia.org/wiki/Computer_science) y [lingüística](https://en.wikipedia.org/wiki/Linguistics) , la jerarquía de Chomsky (a veces denominada jerarquía de Chomsky-Schützenberger ), es una [jerarquía](https://en.wikipedia.org/wiki/Containment_hierarchy) de [contención](https://en.wikipedia.org/wiki/Containment_hierarchy) de clases de [gramáticas formales](https://en.wikipedia.org/wiki/Formal_grammar) .

Esta jerarquía de gramáticas fue descrita por [Noam Chomsky](https://en.wikipedia.org/wiki/Noam_Chomsky) en 1956. También lleva el nombre de [Marcel-Paul Schützenberger](https://en.wikipedia.org/wiki/Marcel-Paul_Sch%C3%BCtzenberger) , quien jugó un papel crucial en el desarrollo de la teoría de [los lenguajes formales](https://en.wikipedia.org/wiki/Formal_language) .

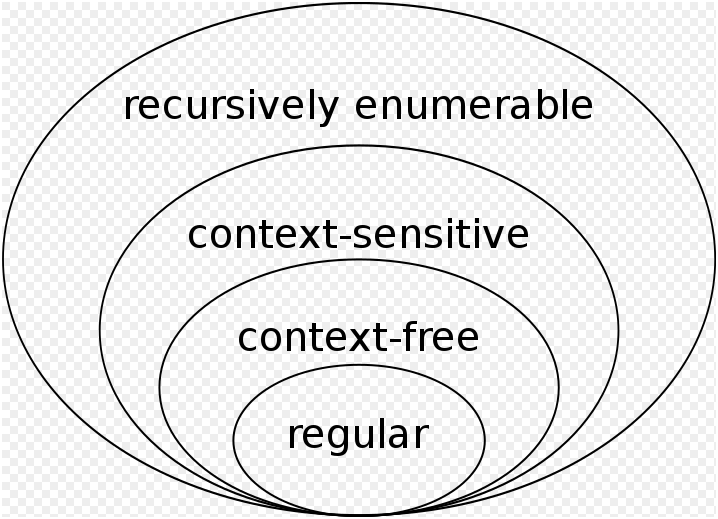


Figura 2.3 jerarquía de Chomsky.

Una gramática formal de este tipo consiste en un conjunto finito de [reglas](https://en.wikipedia.org/wiki/Formal_grammar#Introductory_example) de [producción](https://en.wikipedia.org/wiki/Formal_grammar#Introductory_example) ( lado izquierdo → lado derecho ), donde cada lado consiste en una secuencia finita de los siguientes símbolos:

un conjunto finito de [símbolos no terminales](https://en.wikipedia.org/wiki/Nonterminal_symbol) (que indica que todavía se puede aplicar alguna regla de producción)

un conjunto finito de [símbolos terminales](https://en.wikipedia.org/wiki/Terminal_symbol) (que indica que no se puede aplicar ninguna regla de producción)

un símbolo de inicio (un símbolo no terminal distinguido)

Una [gramática formal](https://en.wikipedia.org/wiki/Formal_grammar) proporciona un [esquema de axioma](https://en.wikipedia.org/wiki/Axiom_schema) para (o genera ) un lenguaje formal , que es un conjunto (generalmente infinito) de [secuencias de símbolos de longitud finita](https://en.wikipedia.org/wiki/String_(computer_science)) que se pueden construir aplicando [reglas de producción](https://en.wikipedia.org/wiki/Production_(computer_science)) a otra secuencia de símbolos (que inicialmente contiene solo la símbolo de inicio). Se puede aplicar una regla reemplazando una aparición de los símbolos en su lado izquierdo con los que aparecen en su lado derecho. Una secuencia de aplicaciones de reglas se denomina derivación . Tal gramática define el lenguaje formal: todas las palabras que constan únicamente de símbolos terminales a los que se puede llegar mediante una derivación del símbolo de inicio.

No terminales son a menudo representados por letras mayúsculas, letras minúsculas por los terminales, y el símbolo de inicio por S . Por ejemplo, la gramática con terminales { a, b } , no terminales { S, A, B } , reglas de producción

S → AB

S → ε (donde ε es la cadena vacía)

A → aS

B → b

y el símbolo de inicio S , define el idioma de todas las palabras de la forma{\ Displaystyle a ^ {n} b ^ {n}}(es decir, n copias de a seguido de n copias de b ).

La siguiente es una gramática más simple que define el mismo idioma:

Terminales { a, b } , No terminales { S } , Símbolo de inicio S , Reglas de producción

S → aSb

S → ε

Como otro ejemplo, una gramática para un subconjunto de juguetes del [idioma inglés](https://en.wikipedia.org/wiki/English_language) viene dada por:

terminales

{generar, odio, genial, verde, ideas, lingüistas}

no terminales

{ S, NP, VP, N, V, Adj }

reglas de producción

S → NP VP

NP → Adj NP

NP → N

VP → V NP

VP → V

N → ideas

N → lingüistas

V → generar

V → odio

Adj → genial

Adj → verde

y empezar símbolo S . Un ejemplo de derivación es

S → NP VP → Adj NP VP → Adj N VP → Adj NV NP → Adj NV Adj NP → Adj NV Adj Adj NP → Adj NV Adj Adj N → great NV Adj Adj N → great linguists V Adj Adj N → grandes lingüistas generan Adj Adj N → los grandes lingüistas generan grandes Adj N → los grandes lingüistas generan una gran N verde → los grandes lingüistas generan grandes ideas verdes.

Otras secuencias que se pueden derivar de esta gramática son: "las ideas odian a los grandes lingüistas " y "las ideas generan ". Si bien estas oraciones no tienen sentido, son sintácticamente correctas. Una oración sintácticamente incorrecta (por ejemplo, " ideas, ideas, gran odio ") no puede derivarse de esta gramática. Ver " [Las ideas verdes incoloras duermen furiosamente](https://en.wikipedia.org/wiki/Colorless_green_ideas_sleep_furiously) " para un ejemplo similar dado por Chomsky en 1957; ver [la gramática estructura de la frase](https://en.wikipedia.org/wiki/Phrase_structure_grammar) y [reglas de estructura de frase](https://en.wikipedia.org/wiki/Phrase_structure_rules) para más [de lenguaje natural](https://en.wikipedia.org/wiki/Natural_language) ejemplos y los problemas de [gramática formal](https://en.wikipedia.org/wiki/Formal_grammar) en esa zona.

1.4 Teoría de autómatas.

La teoría de autómatas es una rama de la [teoría de la computación](https://es.wikipedia.org/wiki/Teor%C3%ADa_de_la_computaci%C3%B3n) que estudia las máquinas abstractas y los problemas que éstas son capaces de resolver. La teoría de autómatas está estrechamente relacionada con la teoría del [lenguaje formal](https://es.wikipedia.org/wiki/Lenguaje_formal) ya que los autómatas son clasificados a menudo por la clase de lenguajes formales que son capaces de reconocer. También son de gran utilidad en la [teoría de la complejidad computacional](https://es.wikipedia.org/wiki/Teor%C3%ADa_de_la_complejidad_computacional).

Un autómata es un modelo matemático para una [máquina de estado finito](https://es.wikipedia.org/wiki/Aut%C3%B3mata_finito) (FSM sus siglas en inglés). Una FSM es una máquina que, dada una entrada de símbolos, "salta" a través de una serie de estados de acuerdo a una función de transición (que puede ser expresada como una tabla). En la variedad común "Mealy" de FSMs, esta función de transición dice al autómata a qué estado cambiar dados unos determinados estado y símbolo.

La entrada es leída símbolo por símbolo, hasta que es "consumida" completamente (piense en ésta como una cinta con una palabra escrita en ella, que es leída por una cabeza lectora del autómata; la cabeza se mueve a lo largo de la cinta, leyendo un símbolo a la vez) una vez la entrada se ha agotado, el autómata se detiene.

Dependiendo del estado en el que el autómata finaliza se dice que este ha aceptado o rechazado la entrada. Si este termina en el estado "acepta", el autómata acepta la palabra. Si lo hace en el estado "rechaza", el autómata rechazó la palabra, el conjunto de todas las palabras aceptadas por el autómata constituyen el lenguaje aceptado por el mismo.

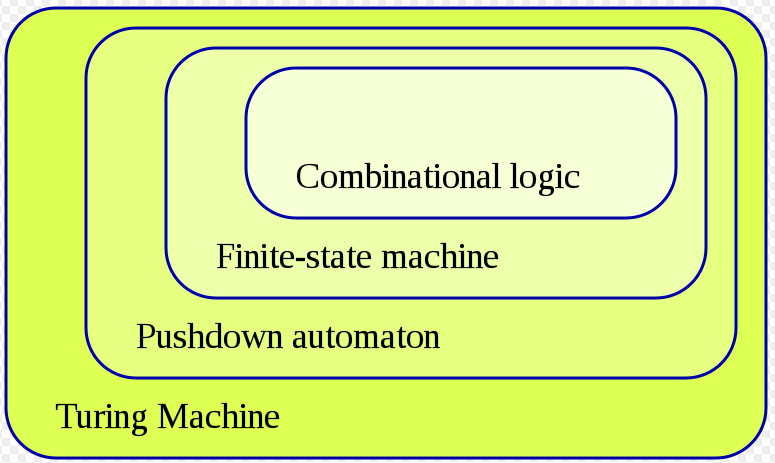


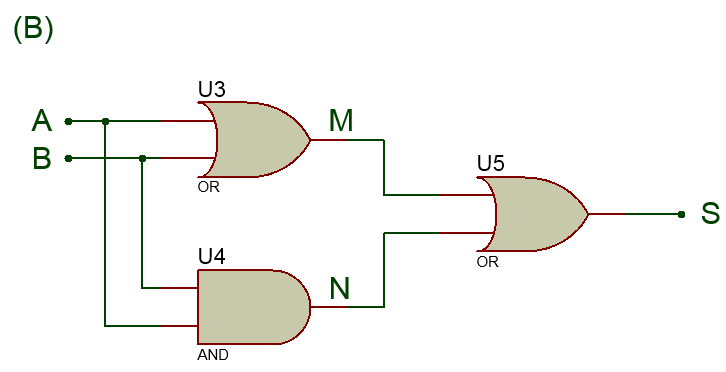
Figura 2.4 Teoría de autómatas.

Capitulo 2. Metodología.

Puerta lógica.

2.1. Ejercicio 1.

Cuando el circuito está formado por varias puertas lógicas, se puede ir calculando paso a paso, nombrando la salida de cada una de ellas para simplificar los pasos. Veamos como hacerlo.



La salida de la puerta lógica OR (U3) está etiquetada como M, mientras que, la salida de la puerta lógica AND (U4) está etiquetada como N. Seguidamente, calculamos el valor de ambas salidas.

M = A + B

N = AB

La salida del circuito lógico se encuentra en la puerta OR (U5), su valor es:

S = M + N

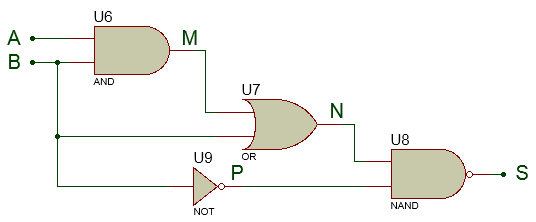
Si ahora sustituimos los valores obtenidos de M y N tenemos la expresión pedida:

**S = A + B + AB**

**Puerta lógica.**

2.2. Ejercicio 2.

En este otro ejemplo, procedemos de la misma manera que en el anterior apartado, etiquetando las salidas de las puertas lógicas que hay en el circuito.



A continuación, obtenemos el valor de cada una de las salidas.

La salida M de la puerta lógica AND (U6) es:

M = AB

mientras que la salida N de la puerta lógica OR (U7) es:

N = M + B

En la puerta NOT (U9), la salida P es:

P = B

finalmente, la salida S es el producto negado de N y P

S = NP = (AB + B) B

Teoría de circuitos.

Ejercicio 1

Sobre un circuito desconocido, que sólo contiene resistencias y fuentes de tensión continua hacemos los siguientes experimentos:

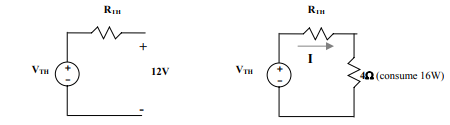
• Conectamos un voltímetro entre dos de sus terminales y observamos que hay una diferencia de tensión de 12V.

• Conectamos una resistencia de 4Ω entre esos mismos terminales y comprobamos que disipa una potencia de 16W.

¿Qué potencia disiparía una resistencia de 2Ω conectada entre los mencionados terminales?

Solucion.

Cualquier circuito puede ser representado por su equivalente Thévenin entre ambos terminales:



Los 12V a circuito abierto se corresponden directamente con VTH: VTH = 12V

La intensidad que recorre el circuito se deduce a partir de la información de potencia:

16W = I2 \*4Ω; I2 = 4A; I = 2A

Y RTH se obtiene a partir de esa intensidad: I = VTH/(RTH+4Ω); RTH + 4Ω = 6Ω; RTH = 2Ω Conocido el equivalente completo se puede obtener el dato pedido:

Con la resistencia de 2Ω: I = 12V/4Ω = 3A P = I2 \*2Ω = 18W

Código de lenguaje de programación c++ que fue posible resolver los casos anteriores de resistencia.

public partial class Form2 : Form

    {

     public Form2()

     {

         InitializeComponent();

     }

     //boton de cerrar

     private void salirToolStripMenuItem\_Click(object sender, EventArgs e)

     {

        Close();

     }

     //boton para calcular Comentario

     private void btncalcular\_Click(object sender, EventArgs e) Evento en el boton

     {

             if (valor1.SelectedIndex.Equals(-1) || valor2.SelectedIndex.Equals(-1) || valor3.SelectedIndex.Equals(-1))

            {

                    MessageBox.Show("Asignar un color a cada banda");

             }

             else

             {

                 string op = valor3.Text;

                 double c1 = double.Parse(txtmostrar1.Text);

                 double c2 = double.Parse(txtmostrar2.Text);

                 double R;

                 switch (op)

                 {

                     case "Negro":

                         R = (((c1 \* 10) + c2) \* 1);

                         txtresultado.Text = R.ToString();

                         txtresultado.Text += " Ω";

                         break;

                     case "Cafe":

                         R = (((c1 \* 10) + c2) \* 10);

                         txtresultado.Text = R.ToString();

                         txtresultado.Text += " Ω";

                         break;

                     //punto decimal = un cero de mas a la derecha

                     case "Rojo":

                            R = (((c1 \* 10) + c2) \* 100) / (1000);

                         txtresultado.Text = R.ToString();

                         txtresultado.Text += " KΩ";

                         break;

                     //entero = misma cantidad de ceros

                     case "Naranja":

                         R = (((c1 \* 10) + c2) \* 1000) / (1000);

                         txtresultado.Text = R.ToString();

                         txtresultado.Text += " KΩ";

                         break;

                     //mas un cero en la izquierda

                     case "Amarillo":

                         R = (((c1 \* 10) + c2) \* 1000) / (100);

                         txtresultado.Text = R.ToString();

                        txtresultado.Text += " KΩ";

                         break;

                     case "Verde":

                         R = (((c1 \* 10) + c2) \* 100000) / (1000000);

                         txtresultado.Text = R.ToString();

                         txtresultado.Text += " MΩ";

                         break;

                     case "Azul":

                         R = (((c1 \* 10) + c2) \* 1000000) / (1000000);

                         txtresultado.Text = R.ToString();

                         txtresultado.Text += " MΩ";

                         break;

                     case "Morado":

                         R = (((c1 \* 10) + c2) \* 10000000) / (1000000);

                         txtresultado.Text = R.ToString();

                         txtresultado.Text += " MΩ";

                         break;

                     case "Gris":

                         R = (((c1 \* 10) + c2) \* 100000000) / (1000000000);

                         txtresultado.Text = R.ToString();

                         txtresultado.Text += " GΩ";

                         break;

                     case "Blanco":

                         R = (((c1 \* 10) + c2) \* 1000000000) / (1000000000);

                         txtresultado.Text = R.ToString();

                         txtresultado.Text += " GΩ";

                         break;

                 }

             }

     }

     //boton de limpiar

     private void btnlimpiar\_Click\_1(object sender, EventArgs e)

     {

         this.valor1.Text = "";

         b1btn.BackColor =Color.LightGray;

         this.valor2.Text = "";

         b2btn.BackColor = Color.LightGray;

            this.valor3.Text = "";

         b3btn.BackColor = Color.LightGray;

         this.tolerancia.Text = "";

         tolbtn.BackColor = Color.LightGray;

         this.txtmostrar1.Text = "";

         this.txtmostrar2.Text = "";

         this.txtmostrar3.Text = "";

         this.txtresultado.Text = "";

     }

     //seleccion de primer color

     private void valor1\_SelectedIndexChanged\_1(object sender, EventArgs e)

     {

         string op = valor1.Text;

         switch (op)

         {

             case "Cafe":

                 txtmostrar1.Text = "1";

                 b1btn.BackColor = Color.Brown;

                 break;

             case "Rojo":

                 txtmostrar1.Text = "2";

                 b1btn.BackColor = Color.Red;

                 break;

             case "Naranja":

                 txtmostrar1.Text = "3";

                 b1btn.BackColor = Color.Orange;

                 break;

             case "Amarillo":

                 txtmostrar1.Text = "4";

                 b1btn.BackColor = Color.Yellow;

                 break;

             case "Verde":

                 txtmostrar1.Text = "5";

                    b1btn.BackColor = Color.Green;

                 break;

             case "Azul":

                 txtmostrar1.Text = "6";

                 b1btn.BackColor = Color.DarkBlue;

                 break;

             case "Morado":

                 txtmostrar1.Text = "7";

                 b1btn.BackColor = Color.Purple;

                 break;

             case "Gris":

                 txtmostrar1.Text = "8";

                 b1btn.BackColor = Color.Gray;

                 break;

             case "Blanco":

                 txtmostrar1.Text = "9";

                 b1btn.BackColor = Color.White;

                 break;

             default:

                    b1btn.BackColor=Color.LightGray;

                 break;

         }

     }

     //seleccion de segundo color

     private void valor2\_SelectedIndexChanged\_1(object sender, EventArgs e)

     {

         string op = valor2.Text;

         switch (op)

         {

             case "Negro":

                 txtmostrar2.Text = "0";

                 b2btn.BackColor = Color.Black;

                 break;

             case "Cafe":

                 txtmostrar2.Text = "1";

                 b2btn.BackColor = Color.Brown;

                 break;

             case "Rojo":

                 txtmostrar2.Text = "2";

                 b2btn.BackColor = Color.Red;

                 break;

             case "Naranja":

                 txtmostrar2.Text = "3";

                 b2btn.BackColor = Color.Orange;

                 break;

             case "Amarillo":

                 txtmostrar2.Text = "4";

                 b2btn.BackColor = Color.Yellow;

                    break;

             case "Verde":

                 txtmostrar2.Text = "5";

                 b2btn.BackColor = Color.Green;

                 break;

             case "Azul":

                 txtmostrar2.Text = "6";

                 b2btn.BackColor = Color.DarkBlue;

                 break;

             case "Morado":

                 txtmostrar2.Text = "7";

                 b2btn.BackColor = Color.Purple;

                 break;

             case "Gris":

                 txtmostrar2.Text = "8";

                 b2btn.BackColor = Color.Gray;

                 break;

             case "Blanco":

                 txtmostrar2.Text = "9";

                 b2btn.BackColor = Color.White;

                 break;

             default:

                 b1btn.BackColor = Color.LightGray;

                 break;

         }

     }

     //seleccion de tercer color

     private void valor3\_SelectedIndexChanged\_1(object sender, EventArgs e)

     {

         string op = valor3.Text;

         switch (op)

         {

             case "Negro":

                 txtmostrar3.Text = "1";

                 b3btn.BackColor = Color.Black;

                 break;

             case "Cafe":

                 txtmostrar3.Text = "10";

                 b3btn.BackColor = Color.Brown;

                 break;

             case "Rojo":

                 txtmostrar3.Text = "100";

                 b3btn.BackColor = Color.Red;

                 break;

             case "Naranja":

                 txtmostrar3.Text = "1,000";

                 b3btn.BackColor = Color.Orange;

                 break;

             case "Amarillo":

                 txtmostrar3.Text = "10,000";

                 b3btn.BackColor = Color.Yellow;

                 break;

             case "Verde":

                 txtmostrar3.Text = "100,000";

                 b3btn.BackColor = Color.Green;

                 break;

             case "Azul":

                 txtmostrar3.Text = "1,000,000";

                 b3btn.BackColor = Color.DarkBlue;

                 break;

             case "Morado":

                 txtmostrar3.Text = "10,000,000";

                 b3btn.BackColor = Color.Purple;

                 break;

             case "Gris":

                 txtmostrar3.Text = "100,000,000";

                 b3btn.BackColor = Color.Gray;

                 break;

             case "Blanco":

                 txtmostrar3.Text = "1,000,000,000";

                 b3btn.BackColor = Color.White;

                 break;

             default:

                 b1btn.BackColor = Color.LightGray;

                 break;

         }

     }

     //seleccion de tolerancia

     private void tolerancia\_SelectedIndexChanged\_1(object sender, EventArgs e)

     {

         string op = tolerancia.Text;

         switch (op)

         {

             case "Cafe 1%":

                 tolbtn.BackColor = Color.Brown;

                 break;

             case "Rojo 2%":

                 tolbtn.BackColor = Color.Red;

                 break;

             case "Naranja 3%":

                 tolbtn.BackColor = Color.Orange;

                 break;

             case "Dorado 5%":

                 tolbtn.BackColor = Color.DarkGoldenrod;

                 break;

             case "Plateado 10%":

                 tolbtn.BackColor = Color.Silver;

                 break;

             case "Sin color 20%":

                 tolbtn.BackColor = Color.LightGray;

                 break;

             default:

                 b1btn.BackColor = Color.LightGray;

                 break;

         }

     }

     private void b1btn\_Click(object sender, EventArgs e)

     {

     }

     private void Form2\_Load(object sender, EventArgs e)

        {

     }

    }//\*\*

}//\*\*

Conclusiones.

* Aprender e investigar es objetivo principal de este informe, para conocer los principios de la materia, así como los temas que fueron explicados anteriormente.
* Se comprendió y explico formas de resolver las operaciones las operaciones gracias al programado que se realizó anteriormente para sacar adelantes los ejercicios.
* Sobre la máquina de Turing se realizó un programa para que haga los cálculos matemáticos y lo resuelva de manera rápida y precisa, al igual que el genio de Alan Turing que creo una máquina para que descifrara los mensajes de los nazis así acortando la guerra y salvando vidas.

Referencias bibliográficas y virtuales.

* <https://es.wikipedia.org/wiki/Jerarqu%C3%ADa_de_Chomsky>
* <https://es.wikipedia.org/wiki/Teor%C3%ADa_de_circuitos>
* <https://riunet.upv.es/bitstream/handle/10251/70631/TOC_0425_06_02.pdf>
* <https://es.wikipedia.org/wiki/Puerta_l%C3%B3gica>
* <https://www.logicbus.com.mx/compuertas-logicas.php>
* https://es.wikipedia.org › wiki › Teoría\_de\_autómatas
* [https://dialnet.unirioja.es › descarga](https://dialnet.unirioja.es › descarga )
* http://coolab.umh.es › circuitos