Universidad Nacional de la Matanza

Departamento:
Ingeniería e Investigaciones Tecnológicas

Cátedra:

Fundamentos de TIC's

(Tecnologías de la Información y la Comunicación)

e-mail: fundamentos_tics@unlam.edu.ar

JEFE DE CÁTEDRA:
Mg. Daniel A. Giulianelli

UNIDAD NRO. 4

INTRODUCCIÓN A LOS CIRCUITOS LÓGICOS

COLABORACIÓN:
Guillermo P. Benéitez
Mabel Cilenti
Artemisa Trigueros
Mónica La Rosa

CICLO LECTIVO:
2011

Universidad Nacional de la Matanza

DEPARTAMENTO DE INGENIERÍA E INVESTIGACIONES TECNOLÓGICAS

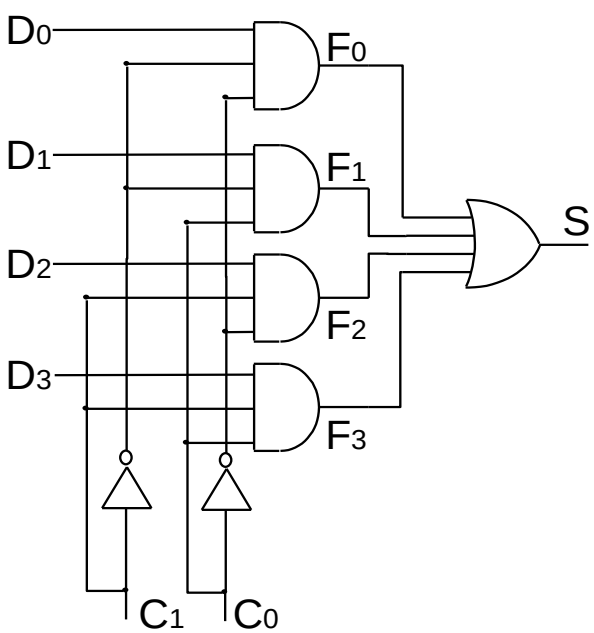
INGENIERÍA EN INFORMÁTICA

FUNDAMENTOS DE TIC´S

UNIDAD 2

Introducción a circuitos lógicos. . .

- Introducción al álgebra conmutacional.
- Demostración de algunos teoremas del álgebra de Boole.
- Implementación de funciones simples.
- Introducción a los circuitos secuenciales.



C_1	C_0	S	Página
0	0	D_0	3
0	1	D_1	13
1	0	D_2	21
1	1	D_3	29

Colaboración: Ing. Guillermo P. Benéitez

Guía general

Página

INTRODUCCIÓN AL ÁLGEBRA CONMUTACIONAL

Conmutación y álgebra conmutacional

Introducción:

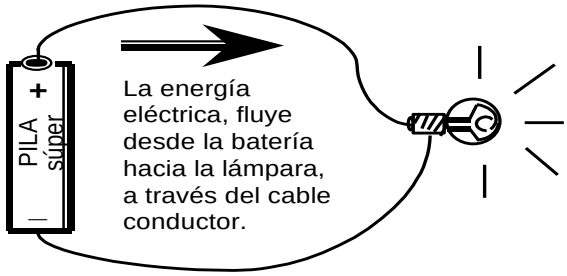
Algunos esquemas mecánicos, circuitos eléctricos, electrónicos y otros sistemas físicos, nos permiten resolver problemas lógicos.

Consideremos algunos parámetros eléctricos básicos[1]:

Elemento	Características	Unidad
Tensión eléctrica	▪ Conocida también como "diferencia de potencial" eléctrico entre dos puntos del espacio en la cercanía de una carga eléctrica. ▪ Es una magnitud física que impulsa a las cargas eléctricas a desplazarse. Fuerza electromotriz. ▪ Producida por un generador eléctrico en un circuito.	Volt (V)
Corriente eléctrica	▪ Consiste en el movimiento de cargas eléctricas. ▪ Entregada por el generador eléctrico en un circuito cerrado. ▪ Corriente eléctrica *continua*: producida por un generador de tensión constante. ▪ Corriente eléctrica *alterna*: producida por un generador de tensión que cambia alternativamente de polaridad. Es la que suministran los generadores que habitualmente se utilizan en los hogares para conectar lámparas y artefactos electrodomésticos.	Ampere (A)
Energía eléctrica	▪ Es la capacidad de producir trabajo que tienen la tensión y la corriente en un circuito. ▪ En corriente continua, resulta del producto de la tensión, por la corriente por el tiempo.	Kilo Watt - hora (Kwh.). (o Joule)
Potencia eléctrica	▪ Es la energía en la unidad de tiempo, es decir, resulta de dividir la energía por el tiempo. ▪ Resulta del producto de la tensión por la corriente.	Watt (W)
Resistencia eléctrica	▪ Es la oposición al paso de la corriente. ▪ Resulta del cociente entre tensión y corriente. ▪ En base al valor de resistencia que adoptan los distintos elementos al hacer circular corriente eléctrica por el circuito, es posible clasificar a los materiales en: *conductores, aisladores y semiconductores*. ▪ Un material es *conductor* cuando no se opone al paso de la corriente eléctrica, lo que significa que es de baja resistencia. Ej.: Cobre (Cu). ▪ Un material es *aislante* cuando se opone al paso de la corriente eléctrica, por lo tanto tiene un elevado valor de resistencia. Ej.: Diamante (C). ▪ Un material se caracteriza como *semiconductor* cuando se comporta como aislador en bajas temperaturas y como conductor a temperatura ambiente (23º C). Ej.: Silicio (Si).	Ohm(Ω)

[1] Gracias por este aporte a Mónica La Rosa.

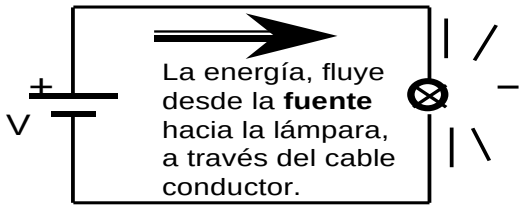
Un circuito eléctrico simple, podrá ayudarnos a entender el concepto.



Representación física

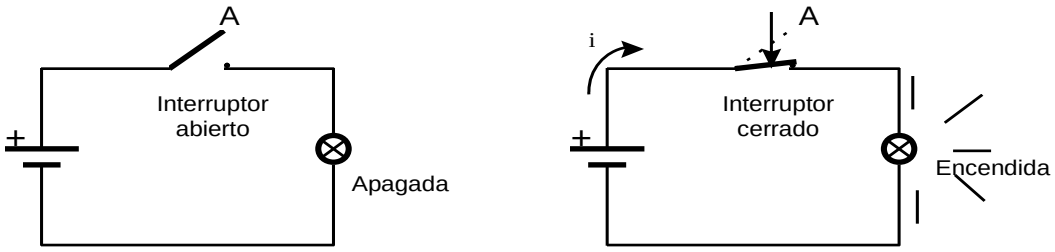
Hemos representado a los conductores eléctricos (cables) mediante trazos curvos.

Este sistema puede simbolizarse por medio de un esquema como el siguiente:



Esquema eléctrico

Para encender o apagar la lámpara, suele emplearse una lámina metálica con un pequeño pliegue que deja al circuito normalmente desconectado[2] en un extremo. Al ejercer presión sobre ella, el extremo abierto se pone en contacto[3] y la corriente eléctrica puede circular. Al dejar de ejercer presión, la lámina metálica se levanta debido a su elasticidad y la corriente se interrumpe nuevamente. A continuación se representan estas situaciones:



Posiciones posibles de un interruptor

El proceso de permitir o no el paso de la corriente eléctrica, por ejemplo cambiando de una posición a otra el interruptor, se denomina **conmutación**[4].

Analicemos la figura anterior:

a) En el circuito de la izquierda, “NO” ejercemos presión sobre el interruptor "A", por lo tanto, la lámpara “NO” se enciende; y
b) En el circuito de la derecha, “SI” ejercemos presión sobre el interruptor "A" y la lámpara “SI” se enciende.

[2] Se denomina interruptor de corriente “Normal Abierto”, o sintéticamente “NA”.
[3] El circuito se cierra.
[4] Obsérvese el comportamiento discreto (digital) del interruptor: conduce o no la corriente eléctrica. Idealmente entre esos estados no hay otros estados intermedios.

Que la lámpara encienda o no, depende de la posición del interruptor "A". Diremos entonces que el encendido de la lámpara es una función de la variable "A".

A partir de este análisis podremos construir la siguiente tabla:

Interruptor "A" presionado	Lámpara Encendida
NO	NO
SI	SI

O más sucintamente:

A	E
NO	NO
SI	SI

También podemos convenir en hacer las siguientes relaciones:

a) NO ejercer presión sobre el interruptor, lo representamos con "0" (cero) y SI ejercer presión sobre el interruptor lo representamos con "1" (uno);

b) lámpara NO encendida: "0" (cero) y lámpara SI encendida "1".

Así, la tabla anterior se verá:

A	E
0	0
1	1

Esta convención se conoce como "Lógica Positiva". Cuando es una relación opuesta a la presente, por ejemplo la lámpara apagada es un "1" y prendida es un "0" se dirá que estamos frente a un sistema que emplea lógica negativa.

El interruptor presionado o no, la lámpara encendida o no, son eventos cotidianos. También son fenómenos físicos evidentes: se toca el interruptor y se "ve" la lámpara.

Otras cuestiones son menos palpables, pero en el fondo tienen las mismas reglas.

Encender la "lámpara del entendimiento" de una persona, dependerá de decirle las palabras adecuadas en el momento apropiado.

La filosofía, la lógica, la educación y las matemáticas son ejemplos de ello.

"... Una teoría matemática (o lógica, en general) es un conjunto de proposiciones que se siguen según un esquema de *deducción lógica*.

Se entiende por proposición, una expresión de la cual tenga sentido inequívoco decir si es verdadera o falsa. ☺

La determinación del criterio de verdad de las proposiciones es a veces una cuestión extra lógica, que pertenece a otro campo del conocimiento.

Por ejemplo: "San Martín murió en Francia", es una proposición cuya verdad pertenece a la Historia. En cambio, decir: "El enfermo morirá o no morirá" es enunciar una proposición verdadera por su misma estructura lógica[5].

En todo proceso de deducción lógica, o *razonamiento*, las proposiciones de partida forman lo que se llama la *hipótesis*, y la conclusión a la que se llega es la *tesis*. ☻

En un razonamiento válido, la tesis *se deduce* o es una *consecuencia lógica* de la hipótesis; también se dice que la hipótesis *implica* la tesis. ..." [6].

☺ Ejemplos de proposiciones:[7]

- Este es un apunte escrito en idioma castellano. (En este caso la proposición es VERDADERA).
- Este es un apunte escrito en idioma chino. (En este caso la proposición es FALSA)

A las tablas empleadas para mostrar los estados de la lámpara y del interruptor se las puede representar según su valor de verdad. Es decir las proposiciones serían:

El Interruptor "A" presionado, la lámpara "E" Encendida.

Cada una de esas proposiciones puede ser Verdadera (V) o Falsa (F), resultando así su Tabla de Verdad.

A	E
F	F
V	V

Observe que "V" corresponde al 1 y "F" al 0 de la tabla anterior.

[5] En este ejemplo "toda la frase" será verdadera en todos los casos posibles, por lo tanto la proposición es siempre verdadera. Toda proposición que resulte siempre verdadera se conoce como "tautología".

[6] Tomado de "ANÁLISIS MATEMÁTICO" de Rey Pastor, Pi Calleja y Trejo; 8ª edición, volumen I, ξ 1 - 2.

[7] Gracias por sus ejemplos y aportes a Artemisa Trigueros.

Se simboliza "P ⇒ Q", para denotar el concepto fundamental de que cuando el razonamiento es válido, la hipótesis P implica la tesis Q.

☻ Ejemplo: [6]

Si el agua está en estado sólido (hielo), entonces su temperatura es menor o igual a 0º Celsius.

Hipótesis (P) ó Antecedente (P) — Tesis (Q) ó Consecuente (Q)

En realidad, estamos afirmando que SIEMPRE que el agua esté en estado sólido, su temperatura es menor o igual a 0º Celsius.
Es muy importante aclarar, que aunque parezca válida toda la frase, el razonamiento que lleva a la conclusión no se aclara en este texto. Seguramente un experto cuestionaría la tesis, indicando que es válida solo si la presión a la que está expuesta la muestra es la normal y otras cuestiones extra lógicas.

En algunos campos del conocimiento, este concepto "P ⇒ Q", se define como *"condicional"* de P a Q.

Evidentemente, **razonando correctamente** es ilógico partir de una hipótesis "P" **verdadera** y llegar a una tesis "Q" **falsa**.

Ejemplo: [6]

Si el agua está en estado sólido (hielo), entonces su temperatura es de 150º Celsius.

Hipótesis (P) ó Antecedente (P) Verdadera (V) — Tesis (Q) ó Consecuente (Q) Falsa (F)

Aquí el razonamiento ha sido falso. No es necesario describirlo pues es un fenómeno observable directamente en la práctica y evidente para el lector.

Las proposiciones pueden ser simples, como:

- Esta materia es Fundamentos de TIC´s.

O compuestas:

- Curso en el turno mañana y viajo en colectivo.

En este ejemplo se emplea la palabra "y" para relacionar (conectar) dos proposiciones simples.
La validez de una proposición compuesta depende de la validez de sus proposiciones simples constitutivas y de cómo estén relacionadas.

Cualquier otra posibilidad puede resultar válida, aún razonando correctamente.
Por ejemplo:

Supongamos la hipótesis compuesta falsa, como ser P: a) la suma de dos enteros es siempre igual al cociente de los mismos **y** b) el cociente de dos enteros es siempre un entero.
Observe que la hipótesis está compuesta de dos proposiciones simples unidas por el "**y**".
Con un razonamiento válido (P ⇒ Q), se deduce que Q: "la suma de dos enteros es siempre un entero". Conclusión (tesis o consecuente) verdadera.

Es importante resaltar que una conclusión acertada donde se ha empleado un razonamiento correcto, no implica necesariamente que la hipótesis sea correcta (como en este y en muchos otros casos).

En todo caso, la conclusión acertada será una condición necesaria, pero no suficiente, como se observa en el ejemplo.

Entonces, cuando logremos demostrar la validez de una hipótesis podremos confiar en la verdad de la tesis obtenida por implicación.

Se pueden emplear tablas para representar las combinaciones posibles de hipótesis verdaderas (V) o falsas (F) y tesis verdaderas o falsas.

Se las denomina "Tablas de Verdad".

Las características de la implicación se pueden aclarar mediante su tabla de verdad:

P	Q	P ⇒ Q
F	F	V
F	V	V
V	F	F
V	V	V

La tabla muestra que la *implicación* de P a Q es falsa únicamente en el caso de que el antecedente P sea verdadero y que el consecuente Q sea falso.

Otra forma usual de leer "P ⇒ Q", sería: *"Si* P*, entonces* Q*"*

En la tabla representamos las cuatro combinaciones posibles[8] de hipótesis y tesis.

Otros ejemplos ayudarán a consolidar este concepto.

Para la primera combinación: P falsa, Q falsa, implicación (P ⇒ Q) verdadera.

Supongamos la siguiente hipótesis compuesta falsa, P: a) la suma de dos enteros es siempre igual al cociente de los mismos y b) el cociente de dos enteros es siempre una fracción.

Se observa que razonando bien (P ⇒ Q), se deduce que "la suma de dos enteros es siempre una fracción". Conclusión Q (tesis) que es falsa.

La segunda combinación que se indica en la tabla de verdad, está ejemplificada en párrafos anteriores, donde: P falsa, Q verdadera, implicación (P ⇒ Q) verdadera.

La tercera combinación es distintiva de la implicación:

P verdadera, Q falsa, implicación (P ⇒ Q) falsa[9].

Para una hipótesis compuesta verdadera, P por ejemplo: a) la tierra es un astro, y b) las estrellas son astros.

Decir que "la tierra es una estrella", no solo es una conclusión Q (tesis) falsa, sino que no fue una deducción lógica de la hipótesis.

Una confusión frecuente acerca de la implicación, proviene de pensar a la primera proposición P, como causa de la segunda Q.

P y Q, pueden no tener nada que ver entre sí, por ejemplo: "Si 2+2=4, entonces hay vida en Marte". Esto representa un absurdo, aunque desde el punto de vista formal, se pueden plantear propuestas como esta.

Observamos que independientemente de la veracidad o no de la expresión "hay vida en Marte", no es una consecuencia lógica de la afirmación "2+2=4".

En algunos casos, P y Q pueden no tener ninguna relación lógica

La última combinación posible:

P verdadera, Q verdadera, implicación (P ⇒ Q) verdadera.

Para una hipótesis compuesta verdadera, como P: a) la tierra es un planeta, y b) todos los planetas son astros.

Se deduce que la tierra es un astro.

En general, los teoremas y experiencias científicas, tienen proposiciones compuestas en su hipótesis, como muestran ejemplos expuestos.

En el contexto de geometría analítica, los enunciados de algunos teoremas, adoptan una forma condicional.

[8] Formadas por las variaciones con repetición de dos objetos V y F con orden igual al número de proposiciones.

[9] Hemos advertido que es ilógica, por eso esta implicación resulta falsa.

Por ejemplo, el teorema de Pitágoras: En un triángulo rectángulo, la suma de los cuadrados de las longitudes de los catetos es igual al cuadrado de la longitud de la hipotenusa.

Hipótesis: El triángulo es rectángulo.

Tesis: La suma de los cuadrados de las longitudes de los catetos es igual al cuadrado de la longitud de la hipotenusa.

Demostrar el teorema, significa establecer la validez de la proposición condicional que constituye su enunciado.

En otras disciplinas, la proposición condicional puede emplearse como elemento de decisión para el control de un proceso.

En esos contextos, la forma *"Si* P*, entonces* Q*"*, incluye una *condición* P y una *orden de control* Q. El condicional dice: "cuando se cumpla la *condición* P, *debe ejecutarse la acción* Q".

Ejemplos:

Si hay luz roja, deténgase.

Si la temperatura del fluido supera los 100º Celsius, entonces abra la válvula de salida de vapor.

Se sugiere buscar otros ejemplos de proposiciones que verifiquen la implicación.

Otras formas de leer la implicación "P $\Rightarrow$ Q", podrían ser: "Q es una condición necesaria para P", y también: "P es una condición suficiente para Q".

Por ejemplo:

P: El entero x, es múltiplo de 4.

Q: El entero x, es par.

La proposición condicional (implicación) "P $\Rightarrow$ Q", puede enunciarse de cualquiera de las siguientes formas:

Si el entero x es múltiplo de 4, entonces es par.

Que el entero x sea múltiplo de 4, es **suficiente** para que sea par.

Que el entero x sea par, es **necesario** para que sea múltiplo de 4.

Tomado del álgebra proposicional, se puede generalizar el nombre "Tablas de Verdad" y emplearlo en todas las tablas de esta obra[10].

Cuando dos proposiciones tienen la misma tabla de verdad, estas son equivalentes.

Álgebra conmutacional:

Existen muchos circuitos distintos con interruptores, algunos de los cuales se presentan muy frecuentemente.

En última instancia, el circuito dependerá de la necesidad concreta.

Por ejemplo para que un ascensor arranque, ambas puertas[11] deben estar cerradas.

También podemos decir que cualquiera de las puertas abiertas deberá impedir el funcionamiento del motor del ascensor.

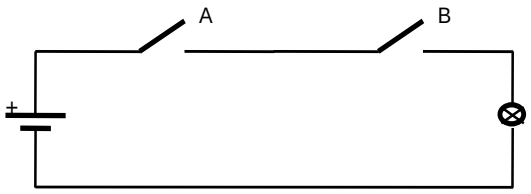
Colocando un interruptor al final del recorrido de cada puerta, los interruptores se abrirán cuando la puerta no esté bien cerrada.

[10] No solo en esta, sino en gran número de obras sobre el tema de sistemas y circuitos digitales o electrónica digital se las llama de ese modo "tablas de verdad".

[11] Una puerta se encuentra en la cabina del ascensor y otra en el piso del nivel en el que está detenido.

El siguiente circuito permite resolver la lógica planteada.

-1) Si colocamos dos interruptores sobre el mismo conductor, es decir, en serie:



Interruptores en serie

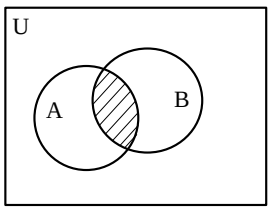
Para que la lámpara encienda (o el ascensor arranque), el interruptor "A" **Y** el "B" deben estar presionados al mismo tiempo. Por lo tanto podrá realizarse la siguiente tabla de verdad:

A	B	E
0	0	0
0	1	0
1	0	0
1	1	1

Independientemente del mecanismo que se emplee, existen muchas situaciones en las cuales, para que un evento esperado se cumpla, deben cumplirse simultáneamente determinadas condiciones.

Como ser, se proponen las condiciones que deben cumplirse simultáneamente para aprobar una materia: aprobación de parciales **Y** cumplir con una asistencia del 75%. En álgebra proposicional recibe el nombre de "conjunción".

Otro caso corresponde a la intersección en la teoría de conjuntos.



La intersección es otro conjunto, el cual está formado por los elementos que pertenecen a uno **Y** al otro simultáneamente.

En la figura adjunta, está representada la intersección entre A y B mediante un diagrama de Euler. Simbólicamente $A \cap B$.

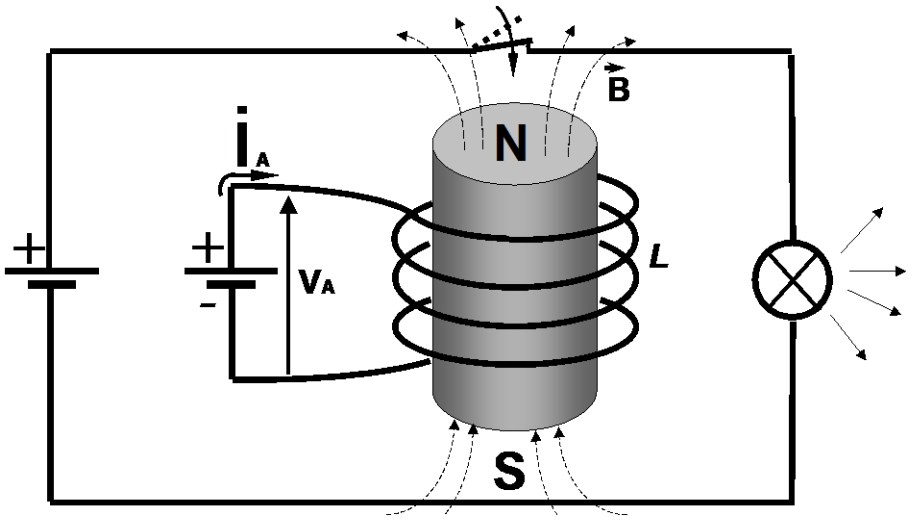
Por estas razones los circuitos que analizamos son llamados **Circuitos Lógicos**.

Esta situación no se presenta solamente en los circuitos eléctricos con interruptores de accionamiento manual, sino también en otras tecnologías.

El mismo resultado se podría lograr utilizando un sistema electromagnético.

En un sistema electromagnético, el interruptor podría estar impulsado por la fuerza de atracción magnética, en lugar de hacerlo por fuerza mecánica (manualmente).

Se muestra a continuación, una representación ilustrativa de lo expuesto.



La lámina elástica que constituye el interruptor (de metal ferroso) es atraída por el campo magnético (**B**). Este está generado por la corriente "$\mathbf{i}_A$", que atraviesa una bobina "***L***" con núcleo de hierro – Este es el principio de funcionamiento de relevadores (Relay) y contactores.-

En este caso, la presión sobre el interruptor "A", estaría originada por la corriente "i_A".

Corriente "i_A"	Lámpara Encendida
NO	NO
SI	SI

O su correspondiente:

A	E
0	0
1	1

La tecnología cambia, pero la estructura lógica es análoga a la anterior.

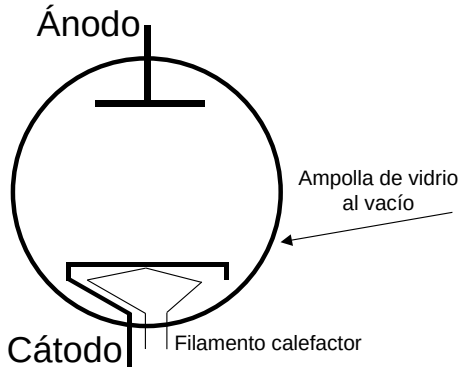
Dispositivos electrónicos

Los dispositivos electrónicos realizan trabajos, controlando corrientes y potenciales eléctricos de circuitos, emisiones de luz y otros efectos, basándose en el control de las cargas eléctricas que constituyen la materia (como electrones).

Dispositivos electrónicos de vacío

Los dispositivos electrónicos de vacío fueron uno de los primeros elementos capaces de controlar electrónicamente el funcionamiento de un sistema.

Se basan en la capacidad de emisión (liberación) de electrones de la superficie de un metal, al elevar su temperatura (emisión térmica) y enfrentarlo a un fuerte potencial positivo (los electrones tienen carga negativa y son atraídos por el potencial opuesto). Este fenómeno se produce dentro de una ampolla de vidrio al vacío (válvulas de vacío).

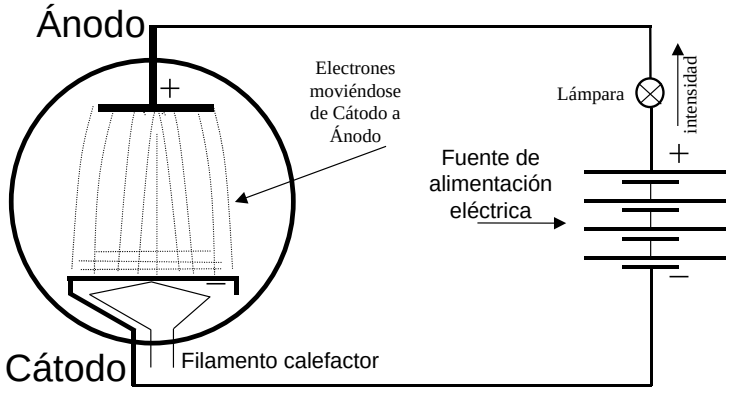


El filamento calefactor conectado a una fuente de energía adecuada, eleva la temperatura (como lo haría una estufa eléctrica), de una placa metálica denominada Cátodo.

Los electrones de la superficie del metal incrementan su velocidad (ya que aumentaron su energía debido a la fuente de calor) y se alejan del Cátodo.

Recordemos que algunos de los electrones que constituyen la materia (átomos, núcleo y electrones), se mueven con cierta libertad por la superficie de los metales.

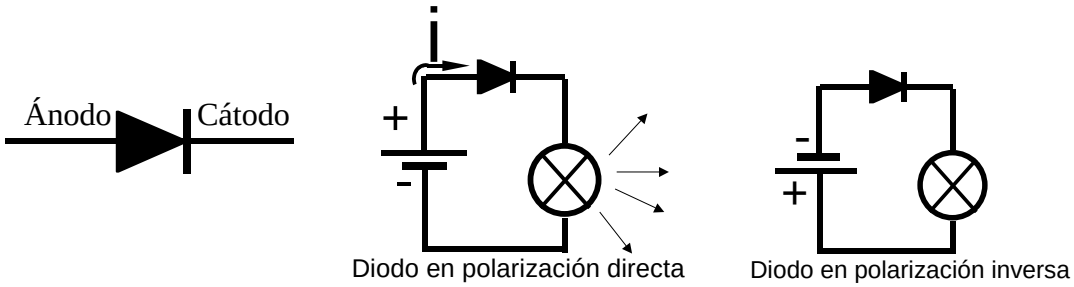
Un elevado potencial eléctrico: negativo en el Cátodo y positivo en otra placa metálica enfrentada (que denominamos Ánodo), permiten alejar los electrones del Cátodo y atraerlos al Ánodo, provocando un movimiento de cargas, es decir, una intensidad de corriente eléctrica en el circuito.



Si en cambio invirtiéramos el potencial eléctrico de la fuente de alimentación (negativo en el Ánodo y positivo en el Cátodo), los electrones (negativos) se verían repelidos por el potencial negativo del Ánodo y no producirían corriente en el circuito.

La válvula de vacío construida de este modo se llama “Diodo”, ó “Diodo valvular”, por su característica de permitir el paso de la corriente en un solo sentido, resultaron de mucha utilidad.

Simbólicamente:



Diodo en polarización directa Diodo en polarización inversa

Existen una gran variedad de válvulas de vacío para distintas aplicaciones: triodos, tetrodos, pentodos, tubos de televisión y monitores de computadoras, microondas, amplificación, etc. .

Dispositivos electrónicos de estado sólido

Muchos dispositivos electrónicos, basan su funcionamiento en materiales semiconductores.

Llamamos conductoras, a las sustancias que permiten el paso de la corriente eléctrica (cobre, hierro, etc.) y aisladores a aquellos que la impiden (amianto, el carbono en forma de cristal: diamante, etc.).

Los elementos como el “silicio” y el “germanio” en forma de cristales, se comportan como aisladores cuando están a bajas temperaturas y como conductores a medida que la temperatura aumenta y, por esto, se los denominó semiconductores.

Existen varias formas de lograr la conducción con semiconductores (aparte de aplicarles calor). Por ejemplo, se los puede hacer conducir aplicándoles luz sobre su superficie (fotoconductividad).

Otra manera de controlar las características de la conducción de un semiconductor es reemplazar un pequeña parte del mismo por otra sustancia, por ejemplo reemplazar algunos átomos del cristal de silicio por átomos de aluminio.

Incorporando impurezas de diferentes tipos, se logran construir cristales basados en semiconductores tales que, se comportan como diodos, transistores (capaces de controlar la corriente de un circuito) y circuitos integrados (que contienen complejas estructuras, equivalentes a grandes cantidades de circuitos con diodos y transistores).

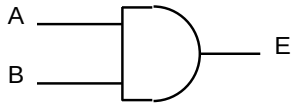
Los dispositivos basados en semiconductores (electrónicos de estado sólido) reemplazan a las válvulas en muchas aplicaciones, sin necesidad de emplear filamentos calefactores internos (ahorrando energía), ocupando menos espacio, disminuyendo las fallas y reduciendo costos, entre otras ventajas.

Existen por lo tanto una gran cantidad de tecnologías para realizar los circuitos lógicos. Algunas emplean válvulas, otras transistores o circuitos integrados, por nombrar solo algunas. En consecuencia, convendrá utilizar una representación que permita esquematizarla, independientemente de la tecnología.

Compuertas lógicas:

Optamos por emplear símbolos conocidos como **Compuertas Lógicas**, que permiten representar operadores lógicos y cuya construcción física podrá realizarse con los dispositivos utilizados en las distintas generaciones de computadores u otros que puedan surgir en el futuro.

Para el caso que nos ocupa, la representación es:



A	B	**E**
0	0	0
0	1	0
1	0	0
1	1	1

Compuerta **Y** (And)

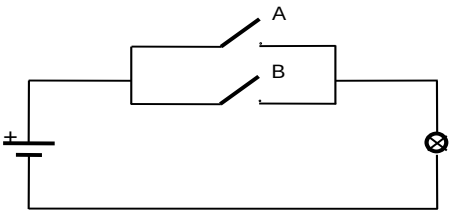
La tabla de verdad se interpreta: para que la salida "E" sea 1, "A **Y** B" deben ser 1.

Esta tabla coincide con el "producto lógico" del álgebra de Boole, donde "**Y**" se reemplaza por el producto lógico representado con un punto "**.**". De esta forma, su tabla de verdad podemos indicarla:

A	B	A . B
0	0	0
0	1	0
1	0	0
1	1	1

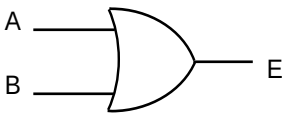
Otra situación diferente, se presenta cuando por ejemplo una alarma debe ser activada desde cualquiera de dos lugares distintos. En este caso presionando uno cualquiera o ambos interruptores debe accionarse la alarma.

-2) Ahora, disponemos dos interruptores en paralelo, es decir, dos líneas (conductores eléctricos) paralelas, cada una de las cuales contiene un interruptor. Esto permite que la lámpara encienda cuando el interruptor "A" **O** el "B" están presionados (indistintamente, incluso pueden estar presionados simultáneamente).



Interruptores en paralelo

Su representación y tabla de verdad se muestran a continuación:



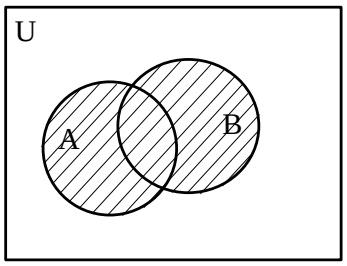
A	B	**E**
0	0	0
0	1	1
1	0	1
1	1	1

Compuerta O (Or)

En teoría de conjuntos está relacionado con la unión de conjuntos.

La unión es otro conjunto, el cual está formado por los elementos que pertenecen a uno **O** al otro.

En la figura adjunta, está representada la unión de A con B mediante un diagrama de Euler. Simbólicamente A ∪ B.



La relación con el álgebra proposicional es a través de la disyunción.

En castellano, la letra "o" presenta ambigüedad respecto del significado lógico.

Si decimos: "esto es blanco o negro", es uno u otro pero no simultáneamente, se excluye la condición de simultaneidad (disyuntiva).

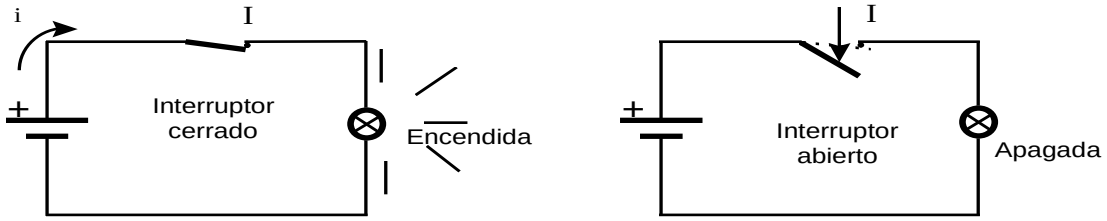
En cambio: “espero tu respuesta, por mail o por teléfono”, implícitamente se acepta la situación de simultaneidad, cualquiera de los medios (mail o teléfono) o los dos al mismo tiempo serán validos. Ahora la “o” es inclusiva[12].

Este último es el sentido en que lo empleamos en este caso y que corresponde a la unión de conjuntos.

La operación lógica O, en álgebra de Boole, equivale a la suma lógica y se representa con el símbolo "+". Así, A **O** B se expresará: A + B (suma lógica de A con B).

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

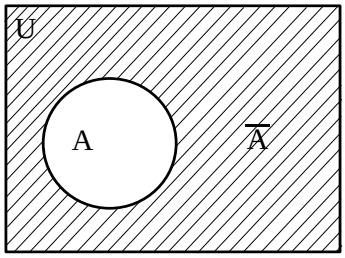
-3) Existe aún un elemento importante en este análisis.



Posiciones posibles de un interruptor inverso al anterior

En el circuito anterior, al ejercer presión sobre el interruptor "I", la lámpara se apaga. Este es un mecanismo inverso al expuesto hasta el momento.

Obviamente existe un elemento equivalente en teoría de conjuntos, el complemento. El complemento de un conjunto es otro conjunto cuyos elementos no pertenecen al conjunto original.

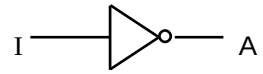


En lógica proposicional, corresponde a "la negación".

Por otra parte, se entiende que este interruptor "I" presenta un funcionamiento opuesto al del interruptor "A", lo cual se indica diciendo: $I = \overline{A}$.

En el álgebra de Boole corresponde a la definición del elemento opuesto.

La compuerta lógica que se emplea para realizar la tarea de invertir una variable y su tabla de verdad se muestran a continuación:

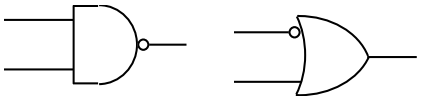


Compuerta Inversora (Not)

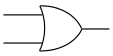
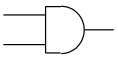
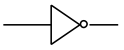
I = no A	A
0	1
1	0

Los inversores, también pueden indicarse por medio de un pequeño círculo entre la variable y el resto del esquema. Por ejemplo, asociándolos a compuertas AND u OR:

[12] En latín, “out - out” es excluyente y “vel” es inclusivo. Un término más preciso sería “yuxtaposición” (en lugar de disyunción), pero no se usa.

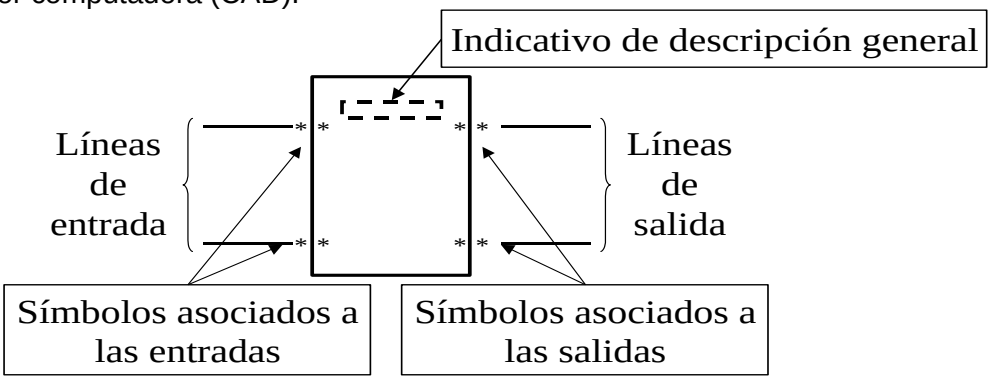


De acuerdo a lo expuesto, realizamos la siguiente tabla, que muestra la relación entre el álgebra de Boole binaria, conjuntos, proposicional, conmutacional y circuitos lógicos.

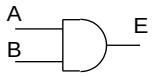
Boole (binaria)	Conjuntos	Proposicional	Conmutación (positiva)	Circuitos lógicos
Elementos	Conjuntos	Proposiciones	Acción, señal	Variables
Suma lógica (+)	Unión ($\cup$)	Disyunción (v), ó	Circuito paralelo	Or 
Producto lógico (.)	Intersección ($\cap$)	Conjunción (^), y	Circuito serie	And 
Elemento opuesto	Complemento	Negación (no)	Inversor	Not 
Neutro de la suma	Conj. Vacío (ϕ)	Falsedad (F)	No a la acción	0
Neutro del producto	Conj. Universal (U)	Certeza (V)	Si a la acción	1

En cuanto a la simbología, conviene aclarar que la representación tradicional que se muestra en este trabajo, es la más empleada en la bibliografía dedicada a profesionales durante la primera etapa de su formación.

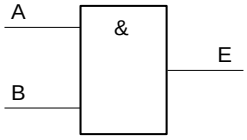
Sin embargo, manuales de datos técnicos y otras publicaciones más avanzadas, utilizan la simbología normalizada (ISO) que se menciona a continuación. Esta última, facilita la representación de bloques funcionales de circuitos integrados digitales y el diseño asistido por computadora (CAD).

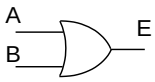


A continuación se muestran algunos ejemplos de la representación normalizada:

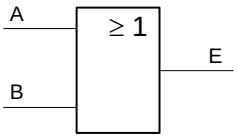
La compuerta AND: 

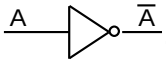
, es equivalente a:



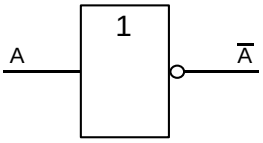
La compuerta OR: 

, es equivalente a:



La compuesta inversora: 

, es equivalente a:



ALGUNAS DEMOSTRACIONES DE TEOREMAS DEL ÁLGEBRA DE BOOLE

Introducción:

Empleando los postulados del álgebra de Boole, demostraremos algunas de las leyes o teoremas[13] necesarios para el desarrollo de circuitos lógicos.

La habilidad en el empleo del álgebra, permite avanzar en la comprensión de las aplicaciones y más tarde redundará en una reducción de costos de los sistemas digitales que, a partir de estas leyes, pueden ser simplificados.

Para que este texto resulte cómodo[14] en el análisis de los teoremas, se transcriben a continuación, los postulados de Huntington.

1) $\exists\ C\ \wedge\ \exists(a, b, ...) \in C\ /\ a\ R\ b\ \wedge\ \exists(a + b) \in C\ \wedge\ \exists(a\ .\ b) \in C$

Recordemos que a y b están en relación de equivalencia "a R b", si se cumplen los principios de identidad "a R a", simetría "a R b $\Rightarrow$ b R a" y transitividad "a R b ^ b R c $\Rightarrow$ a R c".

2 a) $\forall\ (a, b) \in C \rightarrow a + b = b + a$

2 b) $\forall\ (a, b) \in C \rightarrow a\ .\ b = b\ .\ a$

3 a) $\forall\ (a, b, c) \in C \rightarrow (a + b) + c = a + (b + c)$

3 b) $\forall\ (a, b, c) \in C \rightarrow (a\ .\ b)\ .\ c = a\ .\ (b\ .\ c)$

4 a) $\forall\ (a, b, c) \in C \rightarrow (a + b)\ .\ c = a\ .\ c + b\ .\ c$

4 b) $\forall\ (a, b, c) \in C \rightarrow (a\ .\ b) + c = (a + c)\ .\ (b + c)$

5 a) $\forall\ a \in C\ \exists\ N1\ /\ a + N1 = a$

5 b) $\forall\ a \in C\ \exists\ N2\ /\ a\ .\ N2 = a$

6 a) $\forall\ a \in C\ \exists\ \overline{a}\ /\ a + \overline{a} = N2\ \wedge\ /\ a\ .\ \overline{a} = N1$ (6 b)

En algunos casos podemos referirnos a ellos por sus nombres, es decir: 1) Definición, 2) Conmutatividad, 3) Asociación, 4) Distributividad, 5) Existencia del elemento neutro y 6) Existencia del elemento opuesto.

Demostraciones

I) Dualidad:

Cada identidad deducida a partir de los postulados del álgebra de Boole, permanece válida si la operación "+" y "." y los elementos "0" y "1" se intercambian entre sí.

Se deduce de la simetría de los postulados con respecto a las dos operaciones: suma lógica y producto lógico; y a los dos elementos neutros: N1 (el "0" lógico) y N2 (el "1" lógico).

Por otra parte el lector puede escribir nuevamente los postulados realizando el cambio propuesto por el teorema de Dualidad y comprobar que vuelve a obtener los postulados originales, cambiados de orden.

II) a + 1 = 1:

$1 = a + \overline{a} = a + \overline{a}\ .\ 1 = (a + \overline{a})\ .\ (a + 1) = 1\ .\ (a + 1) = a + 1$

Hemos utilizando los postulados: el recíproco de 6a, 5b, 4b, 6a y 5b, respectivamente.

Empleando el concepto de dualidad, se podrá comprobar el teorema: " a . 0 = 0 ", aplicando los postulados: recíproco de 6b, 5a, 4a, 6b y 5a, respectivamente.

[13] En algunos textos se los denomina también "Principios".

[14] Significado de algunos símbolos: $\exists \rightarrow$ existe, ^ $\rightarrow$ y, $\in \rightarrow$ pertenece, / $\rightarrow$ tal que, $\forall \rightarrow$ para todo, $\Rightarrow \rightarrow$ implica.

III) Unicidad: $a + a = a$

$a = a + 0 = a + a \cdot \bar{a} = (a + a) \cdot (a + \bar{a}) = (a + a) \cdot 1 = a + a$

Hemos utilizando los postulados: elemento neutro de la suma, producto con el elemento opuesto, distributividad, suma del elemento opuesto y elemento neutro del producto, respectivamente. También aquí podrá utilizarse el concepto de dualidad y demostrar: " $a \cdot a = a$ ".

IV) Absorción: $a + a \cdot b = a$

$a = 1 \cdot a = (1 + b) \cdot a = (a \cdot 1 + b \cdot a) = a + a \cdot b$

Hemos utilizando los postulados: elemento neutro del producto, teorema "II " aplicado a la variable b, distributividad y producto con el elemento opuesto, respectivamente.

V) Doble negación: $\bar{\bar{a}} = a$

V a) Sea $\bar{a} = x$ y por lo tanto $\bar{\bar{a}} = \bar{x}$.

V b) Considerando el postulado de existencia del elemento opuesto:

$x + \bar{x} = 1$ y $x \cdot \bar{x} = 0$.

V c) Sustituyendo las variables: $\bar{a} + \bar{\bar{a}} = 1$ y $\bar{a} \cdot \bar{\bar{a}} = 0$.

V d) Aplicando el postulado de conmutatividad: $\bar{\bar{a}} + \bar{a} = 1$ y $\bar{\bar{a}} \cdot \bar{a} = 0$.

V e) Que por supuesto satisfacen el postulado de existencia del elemento opuesto:

6 a) $\forall a \in C \; \exists \; \bar{a} \; / \; a + \bar{a} = N2 \; \wedge / \; a \cdot \bar{a} = N1$ (6 b)

Donde N1 = 0 y N2 = 1 y, por lo tanto esa parte del postulado podría escribirse:

6 a) $a + \bar{a} = 1 \; \wedge / \; a \cdot \bar{a} = 0$ (6 b) .

Ya que por el postulado 1), todo elemento del álgebra de Boole es equivalente a sí mismo, y por lo tanto $\bar{a} = \bar{a}$, considerando la deducción del punto V d):

$\bar{\bar{a}} + \bar{a} = 1$ y $\bar{\bar{a}} \cdot \bar{a} = 0$, y el postulado 6 :

$a + \bar{a} = 1 \; \wedge / \; a \cdot \bar{a} = 0$, se deduce que $\bar{\bar{a}} = a$. Como queríamos demostrar.

VI) De Morgan: $\overline{a + b} = \bar{a} \cdot \bar{b}$

VI a) Sea $a + b = u$ y por lo tanto " $\overline{a + b} = \bar{u}$ ", **si se cumple la igualdad presentada en el teorema de De Morgan, entonces se cumplirá:** " $\bar{a} \cdot \bar{b} = \bar{u}$ ". Lo demostraremos empleando el postulado de existencia del elemento opuesto.

VI b) Consideremos ese postulado, aplicado a la variable "u":

$u + \bar{u} = 1$ y $u \cdot \bar{u} = 0$

VI c) Sustituyendo las variables: $(a + b) + (\bar{a} \cdot \bar{b}) = 1$ y $(a + b) \cdot (\bar{a} \cdot \bar{b}) = 0$

VI d) Valiéndonos del postulado de distributividad: $(a + b + \bar{a}) \cdot (a + b + \bar{b}) = 1$

y $(a \cdot \bar{a} \cdot \bar{b}) + (b \cdot \bar{a} \cdot \bar{b}) = 0$

VI e) Ahora aplicamos el postulado de conmutatividad:

$(a + \bar{a} + b) \cdot (a + b + \bar{b}) = 1$

y $(a \cdot \bar{a} \cdot \bar{b}) + (b \cdot \bar{b} \cdot \bar{a}) = 0$

VI f) A continuación utilizamos el postulado de existencia del elemento opuesto:

$(1+ b) . (a + 1) = 1$

y $(0 . \bar{b}) + (0 . \bar{a}) = 0$

VI g) A continuación utilizamos el teorema demostrado en II) "a + 1 = 1" y "a . 0 = 0", aplicados a las variables a ó b, según corresponda:

$(1+ b) . (a + 1) = 1$ ➔ $1 . 1 = 1$

y $(0 . \bar{b}) + (0 . \bar{a}) = 0$ ➔ $0 + 0 = 0$. Como queríamos demostrar.

Funciones de un álgebra de Boole

Una función del álgebra de Boole es una variable binaria[15], cuyo valor depende de una cierta combinación de valores relacionados por las operaciones producto y suma, y donde las variables que intervienen pueden presentarse en forma directa o por medio de su opuesto. Una forma de nombrarla es: $f_{(a, b, c, ...)}$

Por ejemplo: $f_{(b, a)} = a + \bar{b} . \bar{a}$

En este caso, la función $f_{(b, a)}$ vale 1 cuando a = 1, o cuando b = 0 y a = 0. También puede decirse que $f_{(b, a)}$ vale cero, solo si b = 1 y a = 0.

Podemos representar estas posibilidades en una tabla de verdad:

b	a	Valor para $f_{(b, a)} = a + \bar{b} . \bar{a}$
0	0	$f_{(0, 0)} = 1$
0	1	$f_{(0, 1)} = 1$
1	0	$f_{(1, 0)} = 0$
1	1	$f_{(1, 1)} = 1$

O más brevemente:

b	a	$f_{(b, a)}$
0	0	1
0	1	1
1	0	0
1	1	1

Teorema Nº: VII)[16] Para toda función $f_{(a, b, c, ...)}$ se verifica:

$f_{(a, b, c, ...)} = a . f_{(1, b, c, ...)} + \bar{a} . f_{(0, b, c, ...)}$

Podemos comprobar este teorema, verificando los valores que adopta la función para las dos posibles condiciones de la variable "a", es decir: para a = 0 y luego para a = 1.

- Para a = 0, $\bar{a} = 1$; por lo tanto en esta condición $f_{(a, b, c, ...)} = f_{(0, b, c, ...)}$.

En el teorema que nos ocupa: $f_{(0, b, c, ...)} = 0 . f_{(1, b, c, ...)} + 1 . f_{(0, b, c, ...)}$

$f_{(0, b, c, ...)} = 0 + f_{(0, b, c, ...)}$

El primer miembro de la derecha es cero por el teorema II, y el segundo es igual a $f_{(0, b, c, ...)}$ por el postulado del elemento neutro del producto.

- Para a = 1, $\bar{a} = 0$; por lo tanto en esta condición $f_{(a, b, c, ...)} = f_{(1, b, c, ...)}$.

[15] Puede adoptar uno de dos valores: "0", ó "1".

[16] En algunos textos se lo conoce formalmente como Teorema de Expansión.

En el teorema: $f_{(1, b, c, ...)} = 1 . f_{(1, b, c, ...)} + 0 . f_{(0, b, c, ...)}$

$f_{(1, b, c, ...)} = f_{(1, b, c, ...)} + 0$

El primer miembro de la derecha es igual a $f_{(1, b, c, ...)}$ por el postulado del elemento neutro del producto, y el segundo es cero por el teorema "II".

Término canónico[17] ***de una función del álgebra de Boole*** *(Minitérminos y maxitérminos)*

Un producto canónico (o minitérmino), es un producto lógico en el que intervienen la totalidad de las variables (literales: a, b, c, ...) que pertenecen a la función.

Por ejemplo, en la función de tres variables: $f_{(c, b, a)}$, el producto "c . b . a" es un producto canónico. Del mismo modo "$c . \bar{b} . \bar{a}$ ", es otro producto canónico ya que posee todas las variables de la función, sean estas en forma directa o negada.

El teorema anterior, se puede emplear para demostrar la validez de la representación de una función del álgebra de Boole como suma de productos canónicos.

El teorema Dual: $f_{(a, b, c, ...)} = (a + f_{(0, b, c, ...)}) . (\bar{a} + f_{(0, b, c, ...)})$, se utiliza para demostrar la validez de la representación de una función del álgebra de Boole como producto de sumas canónicas.

Una suma canónica, es una suma lógica en la que intervienen la totalidad de las variables (literales: a, b, c, ...) que pertenecen a la función. Por ejemplo, "$c + \bar{b} + \bar{a}$ " en una función de tres variables, es una suma canónica (un maxitérmino).

Representación de una función como suma de productos canónicos y como producto de sumas canónicas

Empleando el teorema VII, podemos analizar una función de la siguiente manera:

$f_{(a, b, ...)} = a . f_{(1, b, ...)} + \bar{a} . f_{(0, b, ...)}$

Aplicamos nuevamente el teorema a los términos de la derecha:

$f_{(a, b, ...)} = a . (b . f_{(1, 1, ...)} + \bar{b} . f_{(1, 0, ...)}) + \bar{a} . (b . f_{(0, 1, ...)} + \bar{b} . f_{(0, 0, ...)})$

Utilizando el postulado de distributividad:

$f_{(a, b, ...)} = a . b . f_{(1, 1, ...)} + a . \bar{b} . f_{(1, 0, ...)} + \bar{a} . b . f_{(0, 1, ...)} + \bar{a} . \bar{b} . f_{(0, 0, ...)})$

Detenemos el análisis aquí, pero queda claro que podría continuarse con las variables c, d, etc.

Supongamos ahora una función de solo dos variables a y b:

$f_{(a, b)} = a . b . f_{(1, 1)} + a . \bar{b} . f_{(1, 0)} + \bar{a} . b . f_{(0, 1)} + \bar{a} . \bar{b} . f_{(0, 0)}$

Ordenando[18] **las variables de mayor a menor (de "Z" a "A"), podremos escribir:**

$f_{(b, a)} = b . a . f_{(1, 1)} + \bar{b} . a . f_{(0, 1)} + b . \bar{a} . f_{(1, 0)} + \bar{b} . \bar{a} . f_{(0, 0)}$

Observe que cuando una variable está directa (no negada) en el lugar que le corresponde en la función hay un 1 y cuando la variable está negada hay un 0

La función quedó expresada como suma de productos. Cada producto es canónico (minitérmino) y está acompañado de una función que puede valer 1 o 0.

Si la función vale 1, ese minitérmino forma parte de la expresión. En cambio si vale 0, el minitérmino desaparece de la misma, ya que al multiplicarlo por 0 el resultado es cero.

El valor de $f_{(b, a)}$, puede tomarse de su tabla de verdad o calcularse con la expresión que la define (aunque no sea canónica).

[17] Existen dos tipos de términos canónicos: producto canónico y suma canónica. Se los conoce en algunos textos como: "términos mínimos" y "términos máximos", respectivamente.

[18] El ordenamiento de las variables, en una forma u otra, no cambia a la función en sí. Es frecuente que cada autor utilice un mismo ordenamiento en toda su obra. En esta, se intenta ilustrar esta cuestión desde el principio de las demostraciones para que el lector pueda manipularlas, independientemente del ordenamiento que tengan. Ahora bien, es sumamente importante conocer qué ordenamiento se emplea, para las representaciones que se explican a continuación.

Por ejemplo[19]: $f_{(b, a)} = b \cdot a \cdot 1 + \bar{b} \cdot a \cdot 1 + b \cdot \bar{a} \cdot 0 + \bar{b} \cdot \bar{a} \cdot 1$; ya que: $f_{(1, 1)} = 1$, $f_{(0, 1)} = 1$, $f_{(1, 0)} = 0$ y $f_{(0, 0)} = 1$.

Quedará: $f_{(a, b)} = b \cdot a + \bar{b} \cdot a + \bar{b} \cdot \bar{a}$

Para facilitar la nomenclatura y el análisis en funciones de mayor número de variables, representaremos cada minitérmino por medio de un número decimal. Este surge de considerar la combinación de unos y ceros de las funciones, como si fueran binarios.

Por ejemplo:

El término que acompaña a $f_{(0, 0)}$, " $\bar{b} \cdot \bar{a}$ " lo representaremos con el número 0.

El término que acompaña a $f_{(0, 1)}$, " $\bar{b} \cdot a$ " lo representaremos con el número 1.

El término que acompaña a $f_{(1, 0)}$, " $b \cdot \bar{a}$ " lo representaremos con el número 2.

El término que acompaña a $f_{(1, 1)}$, " $b \cdot a$ " lo representaremos con el número 3.

De los 4 valores posibles, nuestro ejemplo solo tiene a los minitérminos: 0, 1 y 3.

Como los minitérminos se suman entre sí para obtener la función, lo indicaremos:

$f_{(a, b)} = \Sigma_2 (0,1, 3)$

El subíndice "2" en la sumatoria, representa el número de variables.

Si la función contara con tres variables, los minitérminos posibles serían 8, considerando las combinaciones posibles de las 3 variables y que cada una puede presentarse en 2 formas, negada o no negada. Con el método empleado, en este caso, representaríamos a cada minitérmino con un número decimal entre 0 y 7. Generalizando diremos 2^n posibilidades, para "n" variables y, el número que representa a cada minitérmino será un valor entre "0" y "$(2^n - 1)$".

Los maxitérminos pueden analizarse de la siguiente forma:

1º) ¿Cuáles son los minitérminos que no pertenecen a la función ($\bar{f}$)?.

En este caso solo el Nº 2 ($\bar{f} = b \cdot \bar{a}$) .

2º) complementándolo ($\bar{\bar{f}}$) y aplicando el teorema de "De Morgan" queda:

$f_{(a, b)} = (\bar{b} + a)$.

Este término es una suma canónica, es decir un maxitérmino y su representación en la nomenclatura acordada será el número decimal "1". $f_{(a, b)} = \Pi_2 (1)$.

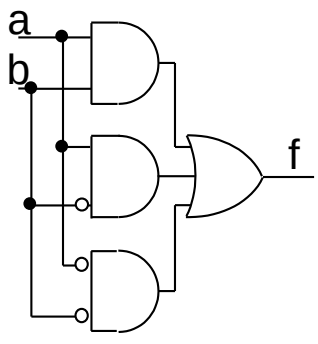
De esta forma, la función tiene dos posibles representaciones canónicas:

$f_{(a, b)} = \Sigma_2 (0,1, 3) = \Pi_2 (1)$.

La primera de las formas " $f_{(a, b)} = \Sigma_2 (0,1, 3)$ ", es la vista anteriormente:

$f_{(a, b)} = b \cdot a + \bar{b} \cdot a + \bar{b} \cdot \bar{a}$

El circuito que le corresponde es:

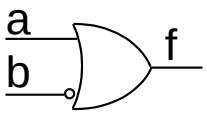


[19] Este ejemplo está tomado de la tabla de la página 16, **Funciones de un álgebra de Boole**. Observe además, que esta función y su tabla de verdad coincide con la "implicación", explicada al definirse las "proposiciones lógicas", p. 4.

La segunda forma " $f_{(a, b)} = \Pi_2 (1)$", que corresponde a la suma canónica es:

$f_{(a, b)} = \bar{b} + a$

Y en esta última, el circuito será:



En ambos casos la función es la misma representada de dos formas distintas pero equivalentes. Resulta obvio que en este caso la segunda resultó mucho más sencilla.

A la hora de realizar el circuito que resuelva una función, será muy importante emplear la versión más simple posible, que además será la más económica, veloz y confiable.

Otro ejemplo aclarará la metodología: Sea la función dada mediante la siguiente tabla:

c	b	a	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Su expresión simbólica será:

$f = \Sigma_3 (2,3,4,5,6,7)$, dada como suma de minitérminos.

Los términos que no pertenecen a la función:

$\bar{f} = \Sigma_3 (0,1)$,

el complemento de esto es: $\bar{\bar{f}} = f = \overline{\Sigma_3 (0,1)}$

y luego aplicando De Morgan será: $f = \Pi_3 (6,7)$.

Observemos que 6 y 7 son los complementos a 7, de 1 y 0 respectivamente.

En general y para facilitar el cálculo de los maxitérminos, el complemento a $(2^n - 1)$ de los minitérminos que no pertenecen a la función, representa a los maxitérminos.

Como práctica sugerimos enunciar la expresión algebraica en sus dos formas.

Por ejemplo los maxitérminos en la función: $f = \Pi_3 (6,7)$, permiten escribir la expresión como producto de sumas canónicas:

$f_{(c, b, a)} = (c + b + \bar{a}) . (c + b + a)$

A raíz de todo lo expuesto, se infiere la necesidad de simplificar las funciones lógicas a fin de lograr circuitos más sencillos y eficientes.

Existen varios métodos de simplificación.

Podremos clasificarlos en tabulares (gráficos) y numéricos.

Los métodos gráficos como los de Veitch o Karnaugh permiten simplificar funciones de pocas variables.

Métodos numéricos como el de Quine - McCluskey, en cambio, suministran una metodología apta para un número cualquiera de variables, aún mucho mayor (>4).

La simplificación, en el caso de suma de productos, intenta reducir la cantidad de variables que intervienen en cada producto. A su vez baja la cantidad de productos. El análisis Dual permite obtener ventajas análogas en el producto de sumas. Eso sí, resultan productos (o sumas) no canónicos.

Los métodos de simplificación se basan en la aplicación de los postulados y teoremas. Para hacerlo algebraicamente, primero se identifican y agrupan los términos donde las variables se repiten salvo una de ellas que, aparece en su forma directa en un término y negada en el otro. Entonces se aplica el recíproco del postulado Nº IV. Luego aplicando el postulado Nº VI se reemplaza la variable agrupada por el término neutro que corresponda y por último se aplica el postulado Nº V.

$f_{(c, b, a)} = (c + b + \bar{a}) . (c + b + a)$

Se podrá agrupar: $f_{(c, b, a)} = (c + b) + (\bar{a} . a) = (c + b) + 0$

$f_{(c, b, a)} = (c + b)$

La variable que desaparece por efecto de la simplificación, tiene un peso tal en el ordenamiento elegido, que coincide con la diferencia entre los valores numéricos decimales asignados inicialmente a cada término canónico de la función.

Por ejemplo, dada la función: $f(c,b,a) = \sum_3(1,3,6,7)$

c	b	a	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$$f = \bar{c}.\bar{b}.a + \bar{c}.b.a + c.b.\bar{a} + c.b.a$$

Aplicamos recíproca de la distributiva (postulado IV) entre los minitérminos 1 con el 3 (diferencia 2, se simplificará la variable b) y 6 con el 7 (diferencia 1, se simplificará la variable a).

$$f = \bar{c}.a.(\bar{b} + b) + c.b.(\bar{a} + a)$$

Aplicamos ahora el postulado del elemento opuesto.

$$f = \bar{c}.a.1 + c.b.1$$

Como "1" es el elemento neutro del producto, resulta:

$$f = \bar{c}.a + c.a$$

El método de Quine – McCluskey, pone en evidencia las ventajas de considerar a los minitérminos en su representación numérica y resultará:

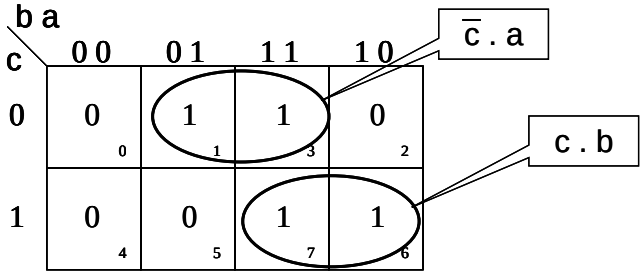
Nº de unos	Término	¿Se utilizó?
0		
1	1	X
2	3	X
	6	X
3	7	X

Pares	Difiere
1 - 3	2
6 - 7	1

	1	3	6	7
1 – 3	x	x		
6 – 7			x	x

Se eliminan las variables cuyo peso corresponde a la diferencia consignada y se deja el resto de las variables: $f = \bar{c}.a + c.a$ Cabe destacar que en este ejemplo sencillo, todos los términos son esenciales primos, de allí que en la tabla final (tabla de implicantes) hayan sido considerados todos los minitérminos de la función. En otros ejemplos más complejos, se verán otras situaciones.

El método de Karnaugh, muestra gráficamente los términos de la función y permite simplificarlos mediante el agrupamiento de los términos adyacentes, en cantidades potencias enteras de 2. Las variables que quedan son aquellas que no cambian.



Para estos y otros métodos, la bibliografía es abundante y se recomienda su lectura comprensiva, estudio y ejercitación correspondiente.

El método tabular de Karnaugh se utiliza luego, en este texto en la última parte de la INTRODUCCIÓN A LOS CIRCUITOS SECUENCIALES, para hallar las expresiones simplificadas de los circuitos que allí se estudian.

Si bien, no es el objetivo de este trabajo analizar el método de Quine - McCluskey, sí nos parece oportuno asentar un ejemplo en las líneas siguientes. Esto permitirá clarificar el tema después de estudiarlo en la bibliografía adecuada.

Se trata de un ejemplo para una función de cuatro variables:

Sea la función $f_{(d, c, b, a)} = \sum_4 (0, 1, 2, 4, 5, 8, 9, 10, 11, 15)$;

En una 1ª tabla, se agrupan en orden creciente de acuerdo a la cantidad de unos que poseen, los términos que pertenecen a la función. Así por ejemplo el 5 (0101_2), está dentro del grupo que tiene el número 2 de unos, junto con el 9 (1001_2) y el 10 (1010_2).

Luego se buscan, entre grupos adyacentes, pares de minitérminos cuya diferencia sea una potencia entera de 2. Como ser el "0" del primer grupo con el "1" del segundo grupo ya que $1 - 0 = 1 = 2^0$. De igual modo entre "0" y "2" ($2 - 0 = 2 = 2^1$) y así con los restantes pares de valores que pueden observarse en esa Tabla 2. A cada término de una tabla que fue utilizado para formar parte de la tabla siguiente, se le coloca una "X" en la fila correspondiente (todos los términos de la primera tabla han sido utilizados en la segunda).

Tabla 1

Cant. de unos	Término	Utilizo?
0	0	X
1	1	X
	2	X
	4	X
	8	X
2	5	X
	9	X
	10	X
3	11	X
4	15	X

Tabla 2

Pares	Difiere	Utilizo?
0 - 1	1	X
0 - 2	2	X
0 - 4	4	X
0 - 8	8	X
1 - 5	4	X
1 - 9	8	X
2 - 10	8	X
4 - 5	1	X
8 - 9	1	X
8 - 10	2	X
9 - 11	2	X
10 - 11	1	X
11 -15	4	(A) *

Tabla 3

Pares de pares	Difiere
0 - 1 - 4 - 5	1 - 4 (B) *
0 - 1 - 8 - 9	1 - 8 (C) *
0 - 2 - 8 - 10	2 - 8 (D) *
0 - 4 - 1 - 5	4 - 1 (E) *
0 - 8 - 1 - 9	8 - 1 (F) *
0 - 8 - 2 - 10	8 - 2 (G) *
8 - 9 - 10 - 11	1 - 2 (H) *
8 - 10 - 9 - 11	2 - 1 (I) *

La Tabla 2 queda ordenada cuando para crearla, se recorre metódicamente la Tabla 1.

En la Tabla 2, han quedado cuatro sectores. Cada sector, está formado por pares de términos tomados de la primera tabla, uno de un bloque y otro del bloque adyacente siguiente.

Otros ejemplos afianzarán el método de razonamiento: el segundo sector, se obtiene al tomar un término del segundo grupo de la Tabla 1 (que tienen solo un uno) y otro del tercer grupo (que tienen dos unos): "1-5", "1-9", "2-10", "4-5" etc., y cuya diferencia también es una potencia entera de 2 ("5 -1" = 2^2, "9 -1" = 2^3, etc.). Luego otro sector entre los que tienen dos y tres unos (9 - 11 y 10 - 11) y un sector (último para este caso particular) entre los que tienen tres y cuatro unos (el 11 - 15). Siempre respetando que las diferencias sean potencias enteras de 2.

Para obtener la siguiente tabla (la Tabla 3), se analizan los sectores adyacentes de la tabla que acabamos de analizar (la Tabla 2 en este caso).

Deben buscarse pares que tengan el mismo valor de diferencia, por ejemplo: en el primer sector, el par: "0 _ 1" difiere en 1 y el par "4 _ 5" del sector adyacente siguiente, también difiere en 1. Ahora se calcula la diferencia entre los pares:

```
   4 _ 5
-  0 _ 1   Se debe verificar que la diferencia entre pares sea una potencia entera de "2"
   4 _ 4        (4 = 2^2)
```

En la tabla 3 se han indicado las diferencias calculadas, por ejemplo: 1 - 4 (B) *. Donde el primer valor: "1" es la diferencia dentro de un par de minitérminos de la Tabla 2 y el segundo valor: "4" es la diferencia entre los dos pares (que acabamos de calcular).

De esta forma se va completando la Tabla 3 hasta terminar de buscar esas condiciones en la totalidad de los valores de la Tabla 2.

Observe que el término A no fue incluido en ningún grupo de la Tabla 3, por esta razón está marcado con un asterisco “*”. Se dice que “A” es un Implicante Primo.

Terminada la Tabla 3, se analizan sus componentes buscando términos adyacentes con diferencias iguales (parejas de diferencias en este caso). Al no haber diferencias iguales entre los dos sectores de esta última tabla, se termina el proceso de agrupación. De lo contrario, debería hacerse una "Tabla 4".

Con el mismo criterio empleado para el término A, los términos: B, C, D, E, F, G, H e I, llevan asterisco “*” (implicantes primos), ya que no fueron incluidos en tablas posteriores.

Algunas agrupaciones contienen los mismos minitérminos, tal es el caso de: B y E ya que incluyen: 0, 1, 4 y 5, solo que en distinto orden y por consiguiente resultan equivalentes y dará lo mismo usar uno u otro. Emplearemos el B. De igual modo los términos: C y F (0-1-8-9), D y G (0-2-8-10) y H e I (8-9-10-11). Emplearemos: C, D y H.

Ahora en la siguiente tabla, incluimos a los minitérminos implicados en la función y las agrupaciones de términos que los contienen y no han podido ser incluidas en agrupaciones de orden mayor (implicantes primos).

La tabla de implicantes (colocamos en ella, solo los términos primos distintos entre sí):

Términos primos		0	1	2	4	5	8	9	10	11	15
A	11 - 15									X	X
B	0 - 1 - 4 - 5	X	X		X	X					
C	0 - 1 - 8 - 9	X	X				X	X			
D	0 - 2 - 8 - 10	X		X			X		X		
H	8 - 9 - 10 - 11						X	X	X	X	
				↑	↑	↑					↑

El análisis de esta tabla nos proveerá de la expresión simplificada de la función.

Existen algunos minitérminos que solo pueden ser resueltos mediante una agrupación determinada, (tienen una sola cruz en su columna y han sido identificados con una flecha en la parte inferior de la taba). Esto nos obliga a emplear esa agrupación, conocida como término esencial. Por ejemplo el minitérmino 15, solo se resuelve en la agrupación “A”.

Términos esenciales son: A: 11 - 15 ($d\ b\ a$); B: 0 - 1 - 4 - 5 ($\overline{d}\ \overline{b}$), y D: 0 - 2 - 8 - 10 ($\overline{c}\ \overline{a}$).

La expresión algebraica que corresponde a cada término esencial se obtiene mediante la simplificación de los minitérminos que le son propios. Veamos uno de ellos, el A:

El minitérmino 11: $d\ .\ \overline{c}\ .\ b\ .\ a$, y el 15: $d\ .\ c\ .\ b\ .\ a$.

La sumatoria es: $d\ .\ \overline{c}\ .\ b\ .\ a + d\ .\ c\ .\ b\ .\ a = d\ .\ b\ .\ a\ .\ (\overline{c} + c) = d\ .\ b\ .\ a$

Otra forma de obtener el resultado correcto es: la diferencia entre los minitérminos indica la variable que se simplifica. Veamos: 15 - 11 = 4. En el ordenamiento de las variables lógicas que hemos empleado: d c b a , y considerando el peso que le corresponde a cada una: $d \rightarrow 2^3$, $c \rightarrow 2^2$, $b \rightarrow 2^1$ y $a \rightarrow 2^0$, la variable que pesa $4 = 2^2$ es “c” y por lo tanto se simplifica (desaparece de la expresión) y el resto de las variables quedan en la condición en la que se encontraban.

De los dos términos no esenciales “C” y “H”, elegimos uno para dar solución al minitérmino "9", ya que los términos esenciales no lo resuelven.

Elegimos “C”: 0 - 1 - 8 - 9 ($\overline{c}\ \overline{b}$). Observe que desaparecieron las variables d y a, por simplificación ya que sus pesos: 8 y 1 coincide con las diferencias numéricas.

0 - 1 Difiere en 1
8 - 9 Difiere en 1
} Entre ellos difieren en 8

La función simplificada queda: $f = d\ b\ a + \overline{d}\ \overline{b} + \overline{c}\ \overline{a} + \overline{c}\ \overline{b}$.

De haber elegido “H”: 8 - 9 - 10 - 11 ($d\ \overline{c}$), (en lugar de “C”) la función quedaría:

$f = d\ b\ a + \overline{d}\ \overline{b} + \overline{c}\ \overline{a} + d\ \overline{c}$.

IMPLEMENTACIÓN DE FUNCIONES SIMPLES

(Circuitos combinacionales)

Introducción

Las compuertas lógicas presentadas en la primera parte "INTRODUCCIÓN AL ÁLGEBRA CONMUTACIONAL", los postulados del álgebra de Boole y los teoremas demostrados en la segunda "ALGUNAS DEMOSTRACIONES DE TEOREMAS DEL ÁLGEBRA DE BOOLE", brindan la posibilidad de analizar y comprender algunos circuitos frecuentemente utilizados en sistemas automáticos de cómputo de datos y control digital. Por otra parte, la simplificación de funciones permite realizarlos de manera eficiente.

Los circuitos combinacionales, proveen una representación gráfica tal que las aplicaciones lógicas y aritméticas, muchas veces resultan más claras.

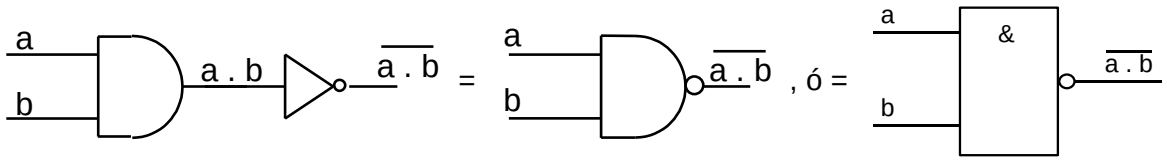
Las funcionalidades de los circuitos combinacionales que se exponen a continuación, se emplean profusamente en aplicaciones prácticas del hardware[20] de sistemas digitales.

Si bien se presentan de manera básica y sencilla, resultan imprescindibles para progresar en materias relacionadas con sistemas de computación.

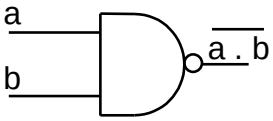
NAND y NOR

Debido a la simplicidad de diseño de circuitos digitales con compuertas del tipo NAND y NOR solamente, varias tecnologías como "lógica de transistor - transistor" (TTL) y otras, propiciaron la construcción de circuitos a partir de ellas.

- La función **NAND** (NO AND), corresponde a la negación del producto lógico. Es decir, se realiza el producto de las variables de entrada y luego se invierte la salida.



Se puede decir que la salida de una compuerta NAND es uno cuando alguna entrada es cero (detecta la presencia de al menos un cero). Desde otro punto de vista, la salida es cero si y solo si todas las entradas valen uno simultáneamente.



b	a	f = $\overline{a \cdot b}$
0	0	1
0	1	1
1	0	1
1	1	0

En una NAND, cuando una de las entradas es 1, la otra se invierte a la salida.

Por ejemplo, si b = 1 → f = $\overline{a}$

Asimismo, uniendo ambas entradas (a = b), la salida será su negado f = $\overline{a}$, ó f = $\overline{b}$.

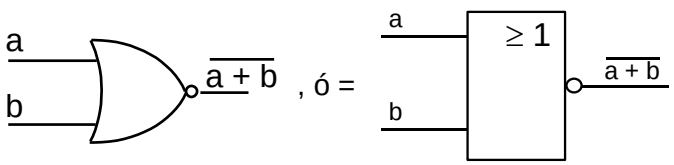
Si colocamos un inversor (negador) a la salida de una NAND obtendremos una AND.

Por otra parte, si negamos cada una de las entradas de una NAND, la función resultante corresponde a una OR. Esto surge de aplicar De Morgan.

- La función **NOR** (NO OR), corresponde a la negación de la suma lógica. Es decir, se realiza la suma de las variables de entrada y luego se invierte la salida.

Se puede decir que la salida de una compuerta NOR es uno cuando todas las entradas son cero simultáneamente (detecta el valor cero en la entrada). Otro enfoque nos permite afirmar que la salida es cero si por lo menos una de las entradas vale uno.

[20] Algunas de estas aplicaciones funcionales se implementan en combinaciones de hardware y software. Más aún, algunas funciones se desarrollan y aplican en sistemas que operan de forma analógica, como los Multiplexores.



b	a	f = $\overline{a+b}$
0	0	1
0	1	0
1	0	0
1	1	0

En una NOR, cuando una de las entradas es 0, la otra se invierte a la salida.

Por ejemplo, si b = 0 → f = $\overline{a}$

Aquí también, uniendo las entradas (a = b), la salida será su negado f = $\overline{a}$, ó f = $\overline{b}$.
Si colocamos un inversor (negador) a la salida de una NOR obtendremos una OR.
Negando cada una de las entradas de una NOR, el resultado corresponde a una AND.

O Exclusiva (Exclusive Or, X-OR)

La "O exclusiva" entrega un uno a su salida, cuando una de sus dos entradas está en uno, pero **no** simultáneamente (se excluye esta condición).

Expresado en forma de función lógica, sería: f = (b + a) . $\overline{b \,.\, a}$

Esto difiere de la operación "O" (inclusiva) definida en el primer postulado del álgebra de Boole en la cual la salida es uno cuando cualquiera de las entradas es uno, aún si todas tienen el valor uno al mismo tiempo.

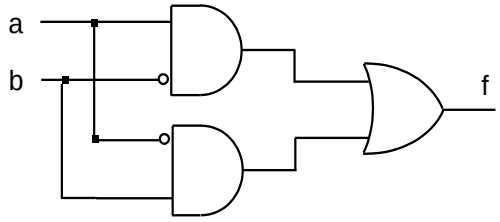
Otra forma de analizar, sería: la salida de una X-OR es uno si una entrada no vale uno y la otra sí, ó viceversa (la primera es uno y la segunda no).

Ahora la expresión sería: f = $\overline{b}$. a + b . $\overline{a}$

Las dos expresiones expuestas son equivalentes y poseen por lo tanto la misma tabla de verdad.

b	a	f
0	0	0
0	1	1
1	0	1
1	1	0

Una de las formas de implementarla mediante compuertas AND, OR e inversoras es:



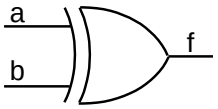
En la tabla de verdad de la X-OR se puede observar que la salida es uno cuando la cantidad de unos de entrada es **"impar"**. Por este motivo se lo emplea como detector de paridad (detecta paridad impar).

Puede, además servir para generar un bit de paridad par. Su salida genera un uno cuando una combinación de entrada es impar y, agregando el bit generado a los bit de entrada, queda un conjunto de bit con paridad par[21].

Otra característica importante es: cuando una de sus entradas es cero (por ejemplo: b = 0), la salida es igual a la otra variable de entrada (f = a) y en cambio, cuando la primera es uno (b = 1), la salida es el opuesto de la segunda (f = $\overline{a}$)[22]. Por este motivo se lo puede utilizar como inversor controlado.

Debido a su importancia y uso extendido en muchas aplicaciones, a la OR exclusiva se le ha asignado el siguiente símbolo:

[21] Note que las filas de la tabla anterior (sus tres columnas) tienen una cantidad par de unos, en todos los casos.
[22] Para razonar este comportamiento, divida la tabla a la mitad y observe que: cuando la entrada "b" es "0", la salida "f" toma los mismos valores que "a", es decir, si a = 0, f = 0 y si a = 1, f = 1. En la parte inferior de la tabla, cuando la entrada "b" es "1", la salida "f" toma los valores opuestos a los de "a", es decir si: a = 0, f = 1 y si a = 1, f = 0.



, y el signo " ⊕ " dentro de las expresiones algebraicas:

f = b⊕a . Se lee: "b OR EXCLUSIVA a".

Multifuciones

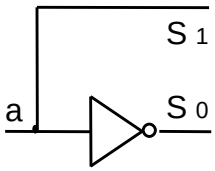
Un sistema que posee varias salidas y por lo tanto responde simultáneamente a varias funciones lógicas, se dice que es una multifunción.

Decodificadores

Un decodificador es un circuito combinacional de varias salidas (multifunción) que convierte la entrada binaria (de uno o más bit), en salidas correspondientes a productos canónicos (minitérminos).

Cada salida tiene, por lo tanto, un solo uno. Este uno se ubica en el minitérmino que coincide con el valor asignado a la salida respectiva (por ejemplo, su equivalente en decimal).

El siguiente circuito lógico, constituye un decodificador elemental. Está formado por una compuerta inversora y una simple línea (o podría darse por medio de una compuerta no inversora). Presenta una entrada "a" y dos salidas "S0" y "S1"

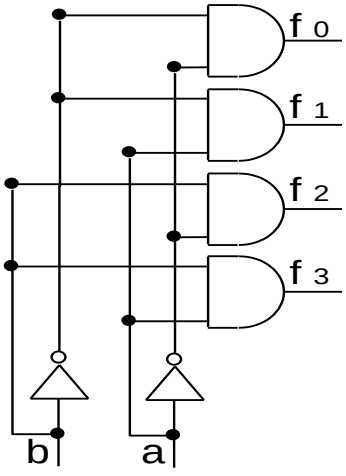


Su tabla de verdad es:

a	S_0	S_1
0	1	0
1	0	1

Las expresiones de las funciones lógicas serán: "$S_0 = \bar{a}$ " y "$S_1 = a$".

Un decodificador un poco más amplio, será el siguiente. Se trata de un decodificador en el cual la entrada está formada por un código binario de dos bit (b a), y cada salida se pone en uno cuando coincide el código binario con el subíndice decimal correspondiente (decodificador binario a decimal).



b	a	f_0	f_1	f_2	f_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

$f_0 = \bar{b} \cdot \bar{a}$

$f_1 = \bar{b} \cdot a$

$f_2 = b \cdot \bar{a}$

$f_3 = b \cdot a$

En la parte inferior del circuito, pueden verse a las entradas "b" y "a", cada una conectada a un decodificador elemental como el analizado en el punto anterior.

Se infiere que, decodificadores de mayores dimensiones, pueden realizarse a partir de circuitos combinacionales simples como, por ejemplo, los expuestos.

Suma aritmética

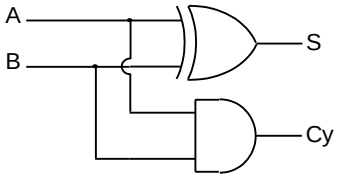
- Un **semi - sumador** aritmético (half adder), es también una multifunción. Cuenta con entradas binarias de un bit "a" y "b" y las salidas: suma "S", y arrastre "C_y" (o acarreo), de acuerdo al siguiente principio:

$$\begin{array}{rr} & a \\ + & b \\ \hline & C_y\ S \end{array}$$

La tabla de verdad se realiza colocando en ella los resultados de la suma binaria: "C_y", y "S", para cada combinación de los valores de "b" y "a".

A continuación se muestra dicha tabla de verdad, el esquema lógico y las expresiones correspondientes a cada salida:

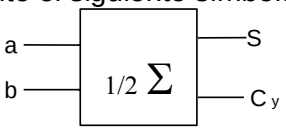
b	a	C_y	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



Se puede observar que el arrastre coincide con el producto lógico "a . b", y la suma aritmética "S" coincide con la función O exclusiva. $C_y = b \cdot a$;

$S = \bar{b} \cdot a + b \cdot \bar{a}$; $S = b \oplus a$

Podemos representar el semi sumador mediante el siguiente símbolo:



Cuando necesitamos sumar números de más de un bit, por ejemplo la suma de dos números de 4 bit: A + B ➔ $a_3\ a_2\ a_1\ a_0 + b_3\ b_2\ b_1\ b_0$, todas las columnas de la suma, con excepción de la primera, requerirán sumar tres bit.

$$\begin{array}{rrrrr} & c_{Y3} & c_{Y2} & c_{Y1} & c_{Y0} \\ & a_3 & a_2 & a_1 & a_0 \\ + & b_3 & b_2 & b_1 & b_0 \\ \hline C_{y\acute{o}} & S_3 & S_2 & S_1 & S_0 \end{array}$$

- El **sumador - total**[23], tiene tres entradas binarias de un bit "a" y "b" y el arrastre de la suma anterior, que lo llamaremos de entrada "c_{Yi}" y las salidas: suma "S", y arrastre "$C_{y\acute{o}}$" al cual nos referiremos como de salida:

$$\begin{array}{rrr} & c_{yi} & \\ & a & \\ + & b & \\ \hline & C_{y\acute{o}} & S \end{array}$$

La tabla de verdad se realiza colocando en ella los resultados de la suma binaria: "$C_{y\acute{o}}$" y "S", para cada combinación de los valores de "c_{Yi}", "b" y "a".

Por ejemplo si:

$c_{Yi} = 0$, b = 0 y a = 0, la suma será: $C_{y\acute{o}} = 0$, y S = 0.
- Ya que 0 + 0 + 0 = 0 y por supuesto no produce arrastre.

$c_{Yi} = 0$, b = 1 y a = 0, la suma será: $C_{y\acute{o}} = 0$, y S = 1.
- Ya que 0 + 1 + 0 = 1 y tampoco produjo arrastre.

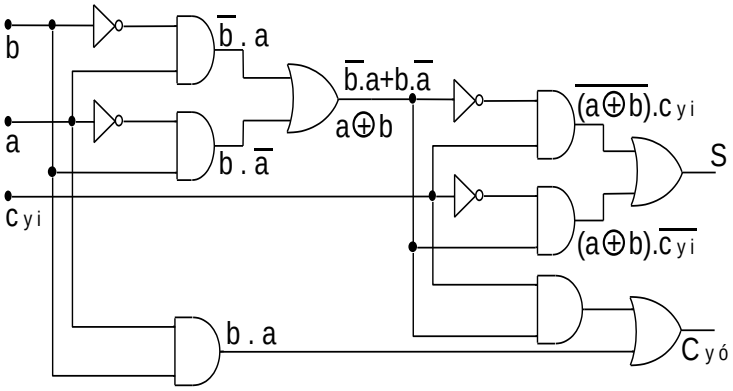
$c_{Yi} = 1$, b = 0 y a = 1, la suma será: $C_{y\acute{o}} = 1$, y S = 0.
- Ya que $1 + 0 + 1 = 10_2$, en binario (2_{10}, en decimal).

$c_{Yi} = 1$, b = 1 y a = 1, la suma será: $C_{y\acute{o}} = 1$, y S = 1.
- Ya que $1 + 1 + 1 = 11_2$, en binario (3_{10}, en decimal).

[23] Full adder.

A continuación se muestra dicha tabla de verdad, el esquema lógico y las expresiones correspondientes a cada salida:

C_{yi}	b	a	$C_{yó}$	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Podemos analizar esta multifunción a partir de su tabla de verdad, considerando la suma de productos canónicos. Para la suma: $S_{(C\,yi,\,b,\,a)} = \Sigma_3\,(1, 2, 4, 7)$

$S = \overline{C_{yi}} \,.\, \overline{b} \,.\, a + \overline{C_{yi}} \,.\, b \,.\, \overline{a} + C_{yi} \,.\, \overline{b} \,.\, \overline{a} + C_{yi} \,.\, b \,.\, a$

Agrupamos aplicando el postulado recíproco de Distributividad:

$S = \overline{C_{yi}} \,.\, (\, \overline{b} \,.\, a + b \,.\, \overline{a}\,) + C_{yi} \,.\, (\, \overline{b} \,.\, \overline{a} + b \,.\, a);$

Aplicando el concepto de X - OR: $S = \overline{C_{yi}} \,.\, (b \oplus a) + C_{yi} \,.\, (\overline{b \oplus a})$;

Volvemos a aplicar X - OR a "C_{yi}" y a "$b \oplus a$":

$$S = C_{yi} \oplus (b \oplus a);$$

Para el arrastre: $C_{yó\,(C\,yi,\,b,\,a)} = \Sigma_3\,(3, 5, 6, 7)$

$C_{yó} = \overline{C_{yi}} \,.\, b \,.\, a + C_{yi} \,.\, \overline{b} \,.\, a + C_{yi} \,.\, b \,.\, \overline{a} + C_{yi} \,.\, b \,.\, a$;

Agrupamos aplicando el postulado recíproco de Distributividad:

$C_{yó} = b \,.\, a \,.\, (\overline{C_{yi}} + C_{yi}) + C_{yi} \,.\, (\overline{b} \,.\, a + b \,.\, \overline{a});$

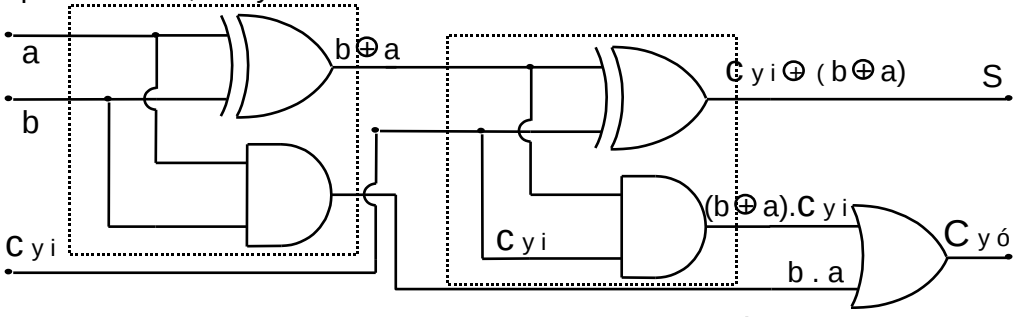
Aplicamos el postulado del elemento opuesto a $\overline{C_{yi}} + C_{yi} = 1$ y el del elemento neutro: $b \,.\, a \,.\, (\overline{C_{yi}} + C_{yi}) = b \,.\, a \,.\, 1 = b \,.\, a$;

Obtenemos: $C_{yó} = b \,.\, a + C_{yi} \,.\, (\overline{b} \,.\, a + b \,.\, \overline{a});$

Aplicando el concepto de X - OR: :

$$C_{yó} = b \,.\, a + C_{yi} \,.\, (b \oplus a);$$

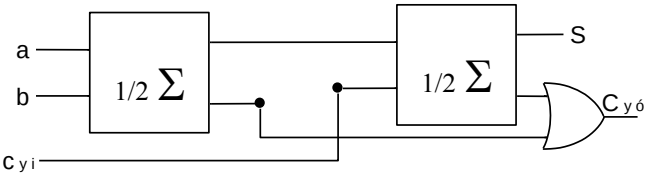
De esta forma, podremos construir el siguiente circuito equivalente al anterior, pero con compuertas AND, OR y X - OR:



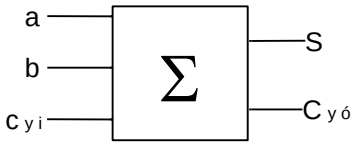
En este esquema podemos descubrir, por simple inspección, la presencia de dos semi - sumadores. El primero suma "a" con "b" y el segundo suma el resultado $b \oplus a$ con C_{yi}.

El arrastre final, por su parte, se obtiene con la suma lógica de los arrastres parciales, es decir será 1 cuando cualquiera de los arrastres parciales sea 1.

Los últimos párrafos nos permiten representar al sumador total mediante:



El sumador total mediante semi sumadores.



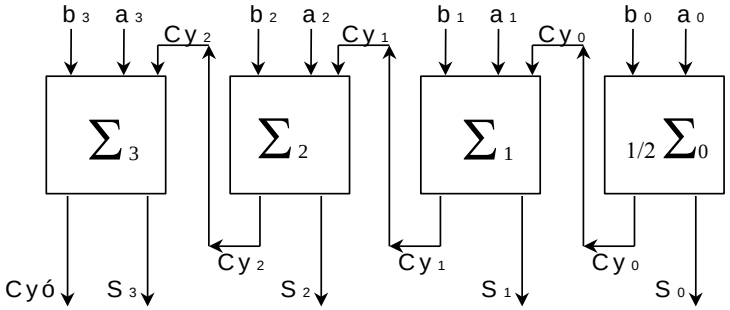
El sumador total como bloque.

Probablemente, este último análisis se acerque a la forma en la que frecuentemente realizamos la operación de suma de tres dígitos. Sumamos los primeros dos y ese resultado lo sumamos al tercero, reservando siempre el arrastre en cada paso.

Para sumar números de varios bit, solemos aplicar la metodología tradicional, que transcribimos y la aplicamos al circuito esquemático simplificado:

	c_{Y3}	c_{Y2}	c_{Y1}	c_{Y0}	
		a_3	a_2	a_1	a_0
+		b_3	b_2	b_1	b_0
	$C_{y\acute{o}}$	S_3	S_2	S_1	S_0

Tomando arrastre en serie:



Comparadores

Un comparador es un sistema que permite establecer si un número es mayor, igual o menor que otro. Se trata, además, de una multifunción.

Básicamente podría razonarse para entradas binarias de un bit "A" y "B" (aunque podrían compararse binarios de "n" bit) y por ejemplo dos salidas: igual "=" ("A = B"), y mayor que ">" ("A>B"). La salida "A es igual a B" se pone en uno cuando las entradas A y B son iguales, y la salida "A mayor a B" se pone en uno cuando el valor de A es mayor que el de B. Podría haber otra salida que informara cuando A es menor que B.

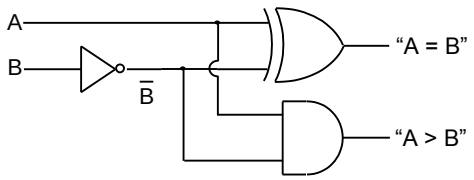
La tabla de verdad de un comparador, se puede deducir a partir el principio expuesto precedentemente:

B	A	A>B	A=B
0	0	0	1
0	1	1	0
1	0	0	0
1	1	0	1

Igualdad	"="	Mayor que	">"
Si A = B	1	Si A>B	1
Si A ≠ B	0	A<B o A = B	0

"A > B" = $A \cdot \overline{B}$

"A = B" = $\overline{A} \cdot \overline{B} + A \cdot B$



El lector podrá deducir la correspondencia entre las expresiones y el esquema lógico, aplicando álgebra de Boole.

El esquema del comparador, nos muestra que contiene un semi - sumador que realiza la suma de las variables en sus entradas, y como la variable B está invertida (complemento a 1), se obtiene la resta mediante la suma del complemento a la base menos uno (de la variable B). Por otra parte este estudio permite reconocer que una comparación no es otra cosa que una resta.

Al restar dos números, el resultado es cero si son iguales. Positivo si el primero es mayor que el segundo y negativo en el caso contrario.

Un concepto importante, es el caso del comparador de igualdad.

B	A	A=B
0	0	1
0	1	0
1	0	0
1	1	1

B	A	$B \oplus A$
0	0	0
0	1	1
1	0	1
1	1	0

Hemos reproducido a la derecha, la tabla de la X - OR.

Se concluye que el opuesto de la X - OR, es un comparador de igualdad. Por este motivo a la X - OR se la conoce también como comparador de desigualdad.

Este efecto, se consigue también negando una sola de las entradas (como se verifica en el circuito anterior, donde se hizo $\overline{B}$).

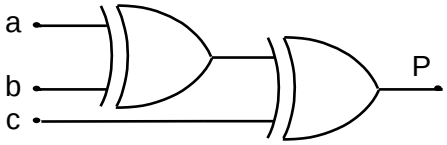
Por lo cual, para la compuerta lógica X - OR se cumple que: $\overline{B \oplus A} = \overline{B} \oplus A = B \oplus \overline{A}$.

Generadores (o detectores) de bit de paridad

Un circuito generador de bit de paridad, permite obtener un bit de paridad par (o impar).

Agregándolo a un código para su transmisión, permitirá detectar los errores posibles de los sistemas teleinformáticos. Los detectores, emplean el mismo concepto para verificar el código recibido. Se trata, en síntesis, de una aplicación de las X - OR.

Por simplicidad, supongamos un código de 3 bit, para el cual queremos generar un bit de paridad par. Esto es, la suma de los unos de un código (incluido el bit de paridad), debe ser par.



c	b	a	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Esta tabla es igual a la suma del Sumador total.

Multiplexores

Un multiplexor es un sistema que permite seleccionar una de varias entradas de datos, mediante un conjunto de líneas de control.

Así, por ejemplo, si tuviéramos varias computadoras y una sola impresora, podríamos seleccionar una de ellas para realizar tareas de impresión, mientras las demás continúan con otros trabajos.

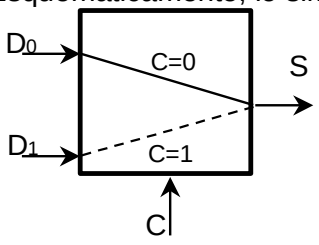
En muchas aplicaciones, aún analógicas, se emplea este concepto. Por ejemplo en un radio grabador, mediante un selector le indicamos al sistema aquello que queremos escuchar: radio o cinta. Incluso seleccionamos radio AM o FM. De esta forma, el sistema electroacústico de salida (los parlantes), quedan conectados a la fuente (datos de entrada) que hemos seleccionado mediante los controles del panel frontal o el control remoto.

Más específicamente, en un sistema digital, y para un sistema con dos posibles entradas de datos, la salida del multiplexor en función de la entrada de control es tal que coincide con la entrada D_0 si el control se encuentra en estado “0” y será igual a D_1 si la entrada de control se encuentra en “1”.

Esquemáticamente, lo simbolizamos:

C	**S**
0	D_0
1	D_1

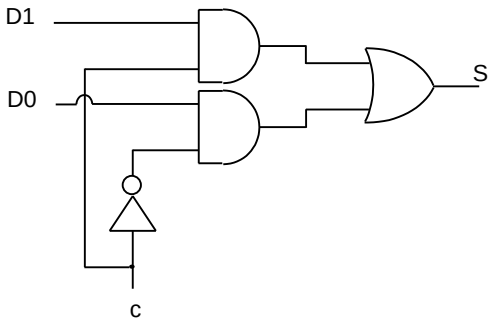
Tabla reducida.



La tabla de verdad y el circuito lógico siguiente, configuran este multiplexor:

C	D_1	D_0	**S**
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Tabla completa.



$S = C.D_1 + \overline{C}.D_0$

Un multiplexor (MUX), puede tener 2^n entradas de datos para "n" terminales de control.

Un MUX de dos entradas de control tiene la siguiente tabla de verdad reducida:

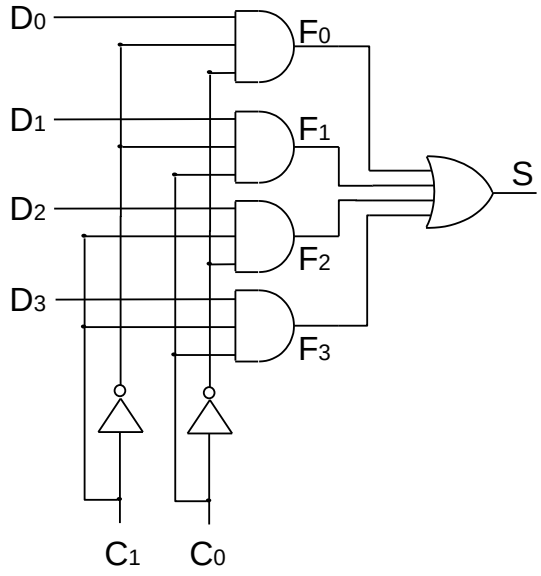
C_1	C_0	S
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

En total, disponemos de 6 entradas (cuatro de datos y dos de control). Si nos proponemos hacer la tabla completa, debemos estar preparados a escribir las 64 combinaciones binarias correspondientes.

De todas formas, la expresión Booleana es fácil de deducir a partir de la tabla reducida:

$$S = \overline{C_1}.\overline{C_0}\ .\ D_0 + \overline{C_1}.C_0\ .\ D_1 + C_1.\overline{C_0}\ .\ D_2 + C_1.C_0\ .\ D_3$$

Un circuito posible es el siguiente:



Las funciones F_0 a F_3 indicadas en el circuito, corresponden a cada uno de los sumandos de la expresión anterior.

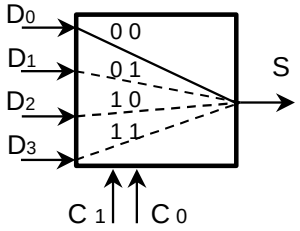
$$F_0 = \overline{C_1}.\overline{C_0}\ .\ D_0$$

$$F_1 = \overline{C_1}.C_0\ .\ D_1$$

$$F_2 = C_1.\overline{C_0}\ .\ D_2$$

$$F_3 = C_1.C_0\ .\ D_3$$

Simbólicamente, podemos dibujar:

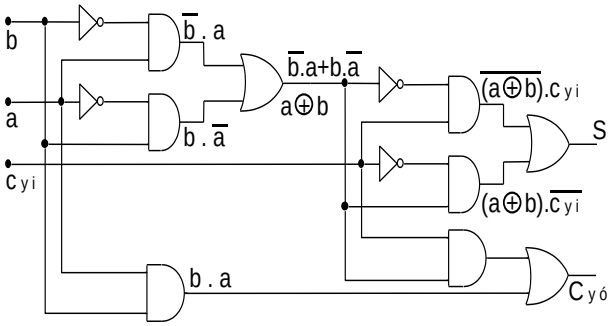


Existen además muchas otras funciones, como los transcodificadores que convierten un código en otro (como por ejemplo la conversión de BCD GRAY a BCD 8421). Algunas funciones inversas a las vistas (como los demultiplexores, que dirigen una entrada de datos a diferentes salidas). Por mencionar solo algunas.

INTRODUCCIÓN A LOS CIRCUITOS SECUENCIALES

En la teoría desarrollada (Introducción a circuitos lógicos), se observó que *un **circuito combinacional** entrega una o varias salidas que responden exclusivamente a una combinación de los valores de las entradas al circuito.*

Por ejemplo:



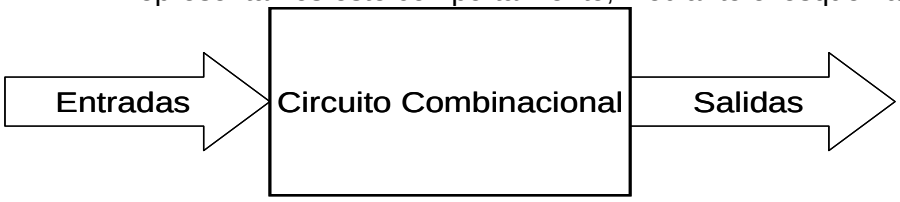
Ejemplo de circuito combinacional: Sumador total ó simplemente Sumador (Full adder)

Aquí, en el sumador, los valores que adoptarán las salidas "S" y "$C_{yó}$", resultan ser una combinación de las variables de entrada "a", "b" y "C_{yi}".

C_{yi}	b	a	$C_{yó}$	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

También podemos decir que a cada combinación de las variables de entrada "a", "b" y "C_{yi}", le corresponde un valor de las salidas "S" y "$C_{yó}$".

Representamos este comportamiento, mediante el esquema genérico:

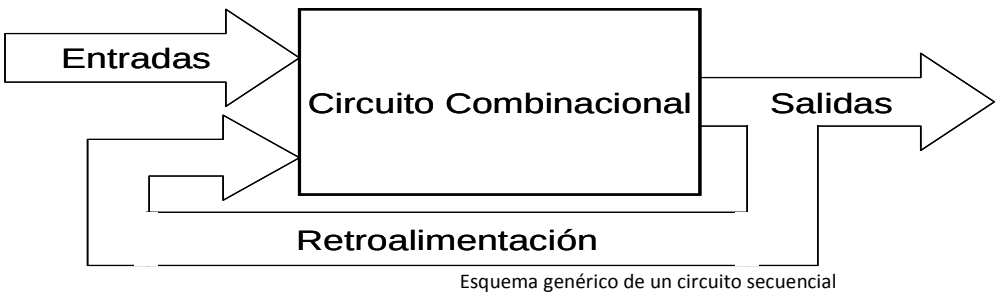


Esquema genérico de un circuito combinacional

En cambio, en un circuito secuencial, las salidas dependen también de la secuencia de valores que fueron adoptando las variables del sistema.

*Un **circuito secuencial** depende del estado de las variables de entrada y de la historia (secuencia) registrada, involucrando el concepto de memoria.*

En un circuito secuencial, se podrá observar un conjunto de compuertas lógicas (como las de un circuito combinacional) y líneas de retroalimentación (feedback) que llevan el valor lógico (estado) de algunas variables hacia la entrada.

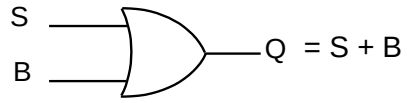


Esquema genérico de un circuito secuencial

Analicemos estos conceptos desde un ejemplo sencillo.

Imaginemos un sistema de alarma para un automóvil. Este deberá accionar la bocina cuando un intruso abra alguna de sus puertas, el baúl etc.

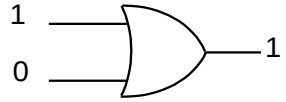
Aprovechando los interruptores dispuestos para el encendido de la luz de cortesía, se puede realizar un sistema que haga sonar la bocina cuando se abra alguna puerta, ó varias.



Esto corresponde a la función lógica "OR", de tantas variables de entrada como aperturas tenga el vehículo. Por ejemplo para dos variables de entrada "S" y "B" (que podrían ser las puertas delanteras), y suponiendo la convención: puerta cerrada "0", puerta abierta "1", bocina sonando "1", quedaría una tabla como la siguiente:

S	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

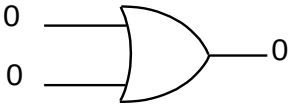
Ahora bien, al abrir la puerta, se activa la alarma



S	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

y el intruso podría salir corriendo cuando escuche el ruido ...

o cerrar la puerta,



en cuyo caso dejaría de sonar.
(El sistema vuelve al estado inicial).

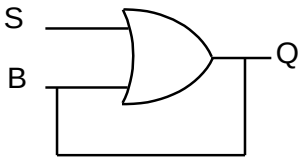
S	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

En estas condiciones sería razonable que, en un segundo intento, el delincuente entre y cierre rápidamente la puerta, perpetrando el delito.

Por este motivo, convendría que el sistema registre la maniobra y mantenga activa su salida (la bocina sonando) permanentemente, a partir de la primera apertura.

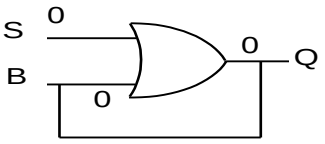
El efecto deseado, puede lograrse conectando la salida de la OR a una de sus entradas. A esta conexión se la denomina "**realimentación**". En este caso, empleamos la entrada "B". De este modo, cuando la salida tome el valor lógico "1", la entrada adoptará ese mismo valor, manteniendo la salida en "1".

B y Q corresponden ahora a la misma conexión (digamos Q). B = Q .



- Observe que ahora, estamos considerando una sola apertura en el vehículo, la "S" -.

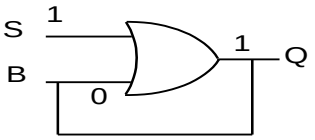
Analizamos el sistema a partir del estado inicial, en que la alarma se encuentra desactivada y la puerta cerrada (todo en "0").



S	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

Al abrir la puerta, la entrada correspondiente ("S") se pondrá en

"1" provocando el cambio de la salida a "1".

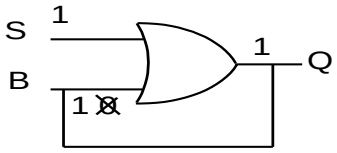


S	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

S	B (Q_i)	Q_f
0	0	0
0	1	1
1	0	1
1	1	1

B y Q difieren momentáneamente durante este estado de transición. Inicialmente "B = 0" (que a partir de ahora lo llamaremos estado inicial Q_i), luego "1" el estado final (Q_f). Al principio Q_i está en "0" y rápidamente cambia a "1" Q_f. Cuando en S se coloca "1", también transcurre un breve lapso (tiempo de demora) hasta que la salida Q adopta ese estado.

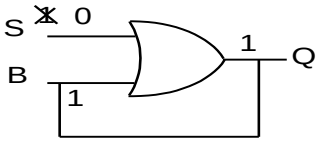
Este cambio de estado en la salida, se reflejará inmediatamente a la entrada, pasando al estado resaltado en la tabla adjunta.



S	B (Q_i)	Q_f
0	0	0
0	1	1
1	0	1
1	1	1

Observe que la compuerta OR presenta "1" en la salida cuando una o más de sus entradas están en "1" por lo tanto, ...

aunque se cierre la puerta ("S = 0") que antes provocó el encendido de la alarma ...



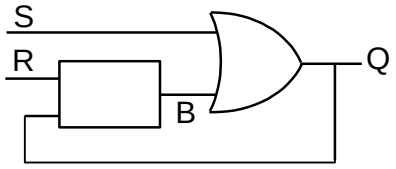
la salida seguirá en estado "1" pues la otra entrada ahora tiene un "1" ("Q =1"). Concepto de memoria.

S	B (Q_i)	Q
0	0	0
0	1	1
1	0	1
1	1	1

S	B (Q_i)	Q_f
0	0	0
0	1	1
1	0	1
1	1	1

El sistema descripto proporciona una solución al problema de encender y mantener la alarma, pero tiene el inconveniente de que una vez activada, seguirá sonando indefinidamente.

Para desactivarla, se podría agregar un circuito lógico en la realimentación, de manera de poder volver al estado inicial. Esto es, poner un cero en el punto indicado como "B = Q" a la entrada de la compuerta OR.

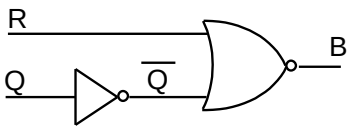


Podemos definir el funcionamiento de ese circuito lógico en la realimentación, diciendo que: a) debe presentar un "0" en los casos en que el usuario desee reiniciar el sistema (R=1, Reset) ó, b) cuando sin resetear (R = 0) la alarma no esté activada (Q = 0). Su tabla es:

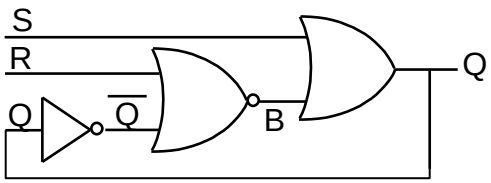
R	Q	B
0	0	0
0	1	1
1	0	0
1	1	0

Corresponde a la función NOR de las variables R con NO Q

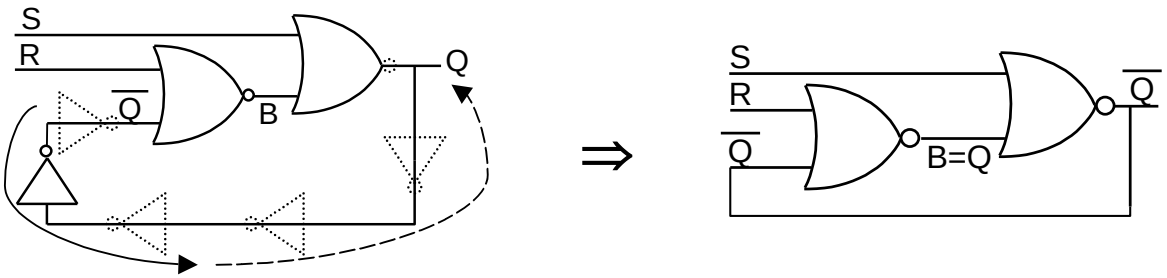
Por lo tanto: $B = \overline{R + \overline{Q}}$



El esquema completo quedaría como se indica a continuación:



Ahora, desplazamos el inversor a través de la línea de realimentación desde la entrada de la NOR hacia la salida de la OR, quedando así, dos compuertas NOR:



Adjuntamos la tabla de verdad completa.
En ella presentamos las dos variables de entrada al sistema R y S, la variable de estado interno B (Q) y la variable de salida Q.
Se observa en el esquema, una salida $\overline{Q}$. La salida original Q, se obtiene del punto B = Q .
Los estados indicados con "X" (estado lógico no relevante), corresponden a la condición en que S y R valgan "1", lo cual representa el absurdo de querer apagar la alarma cuando intentan robar la unidad.

S	R	$B(Q_i)$	Q_f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	X
1	1	1	X

En esta tabla, se presenta el caso de que pueden existir estados no definidos.

En general, si una función presenta estados no definidos y se puede elegir libremente el estado lógico resultante, diremos que se tratará de una "**Función Incompleta**".

Analicemos brevemente la tabla de verdad.

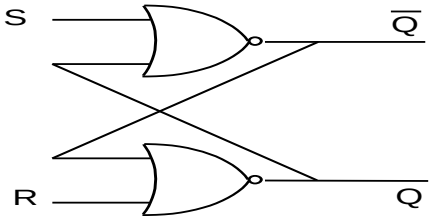
En las dos primeras combinaciones (filas), el estado final de la salida Q_f coincide con el estado anterior (inicial) Q_i . Diremos que el circuito retiene (almacena o memoriza) el estado binario que tenía, sin alterarlo.

En las dos combinaciones siguientes, el estado final de la salida Q_f , es cero independientemente del estado anterior Q_i . Esta es la operación de Reset. Esto se logra gracias a que la entrada R se encuentra en uno y la S en cero.

Las siguientes dos, Q_f está en uno, sin importar el estado en el que se hallaba antes. En este caso lo llamamos Set. Para "setear" (poner en uno) la salida del circuito, debemos asegurar un uno en la entrada Set y un cero en el Reset.

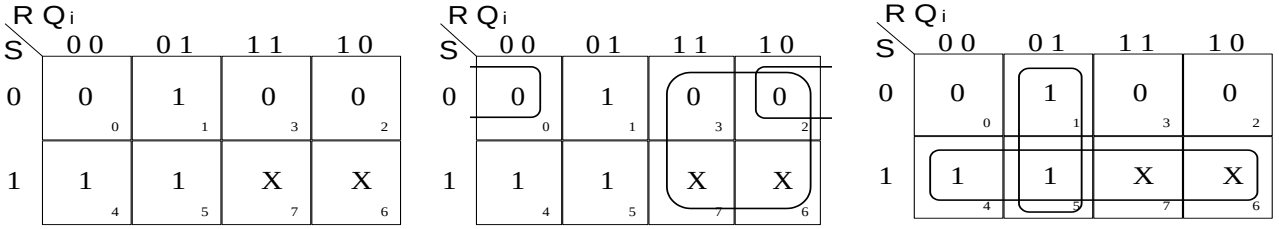
Por ello, al circuito secuencial así formado se lo conoce como **biestable R S**, reset - set, ya que **posee dos estados estables**. También se lo llama **flip flop R S**. Por definición los flip flop, poseen dos salidas: Q y $\overline{Q}$ que se obtuvieron aquí, de la salida de cada una de las NOR.

Por esta razón, en la mayoría de los textos, la forma en la cual se lo encuentra dibujado es:



El último par de combinaciones, se prohiben ya que además de lo expuesto en párrafos anteriores, en este caso impiden el cumplimiento de la definición del flip flop e incluso pueden producir efectos no deseados en su funcionamiento.

En el diseño por métodos tabulares, se colocan en los estados no definidos, una "X". Así el diseñador podrá elegir entre "0" y "1", resultando un circuito más simple. En las siguientes figuras, se muestran las tablas de Karnaugh para la simplificación.



En la segunda tabla, para simplificar por medio de maxitérminos, se ha supuesto que las "X" adoptan el valor cero. La expresión simplificada es: $Q_f = (S + Q_i) \cdot \overline{R}$.

Aplicando el teorema de doble negación y luego De Morgan:

$Q_f = \overline{\overline{(S + Q_i) \cdot \overline{R}}}$; $Q_f = \overline{\overline{(S + Q_i)} + \overline{\overline{R}}}$ y finalmente: $Q_f = \overline{\overline{(S + Q_i)} + R}$ (todas NOR)

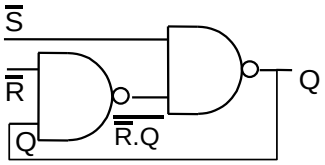
Esta última expresión coincide con el último circuito de la página anterior[24].

En la tercera tabla en cambio, para simplificar por medio de minitérminos, se ha supuesto que las "X" adoptan el valor uno. La expresión simplificada es: $Q_f = S + \overline{R} \cdot Q_i$

Aplicando el teorema de doble negación y luego De Morgan:

$Q_f = \overline{\overline{S + \overline{R} \cdot Q_i}}$, de esta forma: $Q_f = \overline{\overline{S} \cdot \overline{\overline{R}} \cdot Q_i}$ (todas NAND)

A esta última expresión le corresponde el circuito siguiente:



Se observa que las entradas Set y Reset están invertidas. Esto implica que para poner en uno la salida, el $\overline{S}$ debe estar en cero y $\overline{R}$ en uno. Para poner en cero la salida, $\overline{R}$ debe esta en cero y $\overline{S}$ en uno. Si ambas entradas $\overline{S}$ y $\overline{R}$ están en uno, el sistema memoriza (mantiene la salida en su estado anterior). Si ambas estuvieran en cero, se dará el estado prohibido.

Después de este aprendizaje, resultará sintético y esclarecedor realizar la siguiente tabla de verdad reducida de un flip flop R S, donde se muestran las cuatro combinaciones posibles de las variables de entrada R y S y la función de salida final Q_f.

S	R	Q_f
0	0	Q_i
0	1	0
1	0	1
1	1	X

Observe que cuando S y R valen "0", la salida Q_f mantiene el valor anterior de Q (Q_i). Cuando S=1 la salida pasa a "1" y cuando R=1 la salida será "0". Esto concuerda con la tabla completa. También queda claro el estado prohibido correspondiente a S y R simultáneamente en 1.

[24] El lector puede deducir directamente del circuito la expresión algebraica y así comprobar la validez de este análisis.

A esta altura de nuestro estudio, podremos reconocer las expresiones algebraicas y explicar el funcionamiento de otros circuitos posibles para un flip flop R S asincrónico[25].

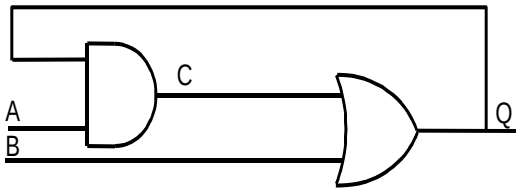
A continuación, se propone un método de análisis.

Consideraremos las características del flip flop R S mediante las cuales: a) se logra poner en uno a la salida con una cierta combinación de las entradas; b) otra combinación logrará poner en cero la salida y c) una tercera combinación mantiene el estado que tenía (memoria de un bit).

Presentado el circuito, se deduce la expresión algebraica. Luego, por medio de las características mencionadas más arriba (o empleando la tabla de verdad característica del flip flop R S) buscamos reconocer las entradas Set y Reset.

En general, veremos el esquema como un sistema con entradas y salidas:

Por ejemplo, el siguiente circuito:



- Recorremos de izquierda a derecha el circuito y encontramos $C = A \cdot Q$
- Continuamos hasta la salida, $Q = B + C$.
- Sustituyendo el valor encontrado para C, en la expresión de Q obtenemos: $Q = B + (A \cdot Q)$.

Ahora bien, como sabemos, la variable de salida realimentada implica que en un momento inicial tendrá un cierto valor y luego un valor final. Ya en párrafos precedentes los hemos indicado con subíndices "i" y "f" respectivamente. Por lo tanto resultará:

$Q_f = B + (A \cdot Q_i)$. Que es la expresión algebraica que representa a este esquema.

Estamos en condiciones de reconocer la función de cada entrada, A y B.

- Observamos que A está conectada a una compuerta AND (producto lógico) y sabemos por los postulados del Álgebra de Boole (más precisamente el "5 b") que el elemento neutro del producto es 1.
- La entrada B, se encuentra conectada a una compuerta OR (suma lógica) y el elemento neutro de la suma es 0.

Un 1 en B pondrá en 1 a la salida, independientemente del estado del resto de las variables.

* Sintéticamente: $B = 1 \Rightarrow Q = 1$. Esto corresponde al Set. Podemos entonces hacer la equivalencia B = S.

Si B está en 0, la salida depende del producto $A \cdot Q_i$, pues: $0 + A \cdot Q_i = A \cdot Q_i$. Si al mismo tiempo, A se encuentra en 1 (neutro del producto) resultará $1 \cdot Q_i = Q_f$.

* Es decir, se conserva el estado anterior (memoria de un bit), con B = 0 y A = 1.

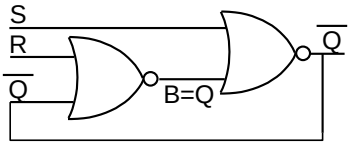
En cambio si B = 0 (neutro de la suma) y A = 0, la salida será cero $Q_f = 0 + 0 \cdot Q_i = 0$.

Sintéticamente: $B = 0$ y $A = 0 \Rightarrow Q = 0$. Esto corresponde al Reset.

* Podemos entonces hacer la equivalencia $A = \overline{R}$, ya que la operación de Reset se logra con A = 0 lo cual corresponde al caso general R = 1 .

[25] Los circuitos secuenciales pueden ser sincrónicos o asincrónicos. Los circuitos sincrónicos necesitan una entrada adicional para sincronizar su funcionamiento. Nos hemos ocupado del tipo asincrónico, porque es más breve su análisis.

Retomando el concepto del biestable RS (Flip Flop RS):



Esquema de un Flip Flop RS

S	R	Q_i	Q_f	
0	0	0	0	Memoria
0	0	1	1	$Q_f = Q_i$
0	1	0	0	Reset
0	1	1	0	$Q_f = 0$
1	0	0	1	Set
1	0	1	1	$Q_f = 1$
1	1	0	X	Estados
1	1	1	X	Prohibidos

Tabla de verdad completa de un Flip Flop RS. Se destacan características de su funcionamiento

En las **dos primeras combinaciones**, el estado final de la salida **Q_f coincide con el estado anterior (inicial) Q_i.** Diremos que el circuito retiene (almacena o memoriza) el estado binario que tenía, sin alterarlo. **S y R en cero**.

En las **dos combinaciones siguientes**, el estado final de la salida **Q_f, es cero** independientemente del estado anterior Q_i. Esta es la operación de **Reset**. Esto se logra gracias a que la entrada **R se encuentra en uno** y un cero en S.

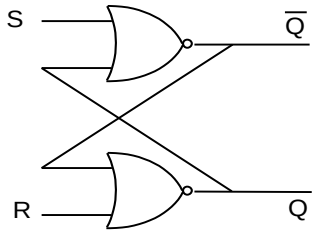
Las **siguientes dos, Q_f está en uno**, sin importar el estado en el que se hallaba antes. En este caso lo llamamos **Set**. Para "setear" (poner en uno) la salida del circuito, debemos asegurar un **uno en la entrada S** y un cero en el R.

Por ello, al circuito secuencial así formado se lo conoce como **biestable R S**, reset - set, ya que su estado posee **dos condiciones estables**. También se lo llama **flip flop R S**.

Por definición: **los flip flop, poseen dos salidas: Q y $\overline{Q}$** que se obtuvieron aquí, de la salida de cada una de las compuertas NOR.

El **último par de combinaciones**, con **S y R en uno**, **se prohíben** ya que en este caso impiden el cumplimiento de la definición del flip flop e incluso pueden producir efectos no deseados en su funcionamiento.

La forma común de dibujarlo es:



Esquema de un Flip Flop RS, en su forma más divulgada

Al diseñar circuitos con combinaciones no definidas, se pueden elegir los valores de la función (cero ó uno), para optimizar el producto final. Es decir, que el circuito resultante tenga la menor cantidad de compuertas que sea posible.

Si empleamos el método de Karnaugh para diseñar el Flip Flop RS suponiendo que los valores de la función para las entradas prohibidas sean "1" (uno), el agrupamiento dará una solución óptima. Se adjunta la tabla correspondiente.

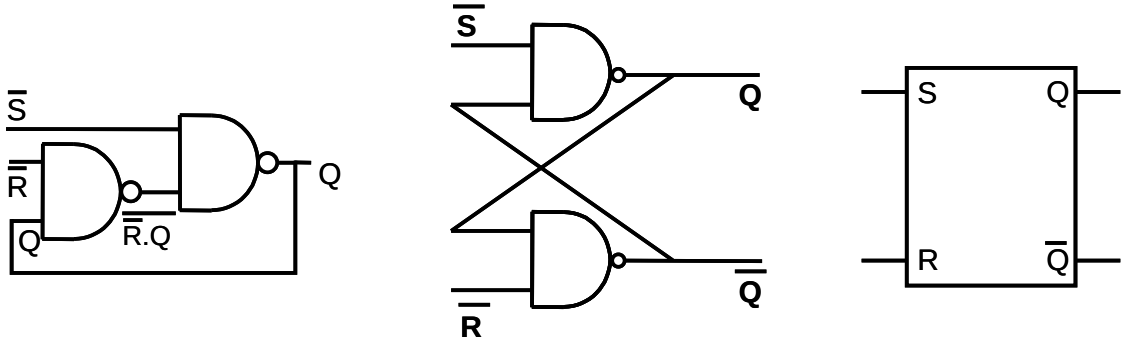
La expresión simplificada es: $Q_f = S + \overline{R} \cdot Q_i$

R Q_i / S	0 0	0 1	1 1	1 0
0	0 (0)	1 (1)	0 (3)	0 (2)
1	1 (4)	1 (5)	X (7)	X (6)

Tabla de Karnaugh, en el análisis del Flip Flop RS

Aplicando el teorema de doble negación: $Q_f = \overline{\overline{S + \overline{R} \cdot Q_i}}$, y luego el teorema de De Morgan: $Q_f = \overline{\overline{S} \cdot \overline{\overline{R} \cdot Q_i}}$.

Este último resultado contiene exclusivamente compuertas NAND.



Dos formas de representar el circuito de un Flip Flop RS con compuertas NAND y su esquema

Advierta que las entradas Set y Reset están invertidas. Esto implica que para poner en uno la salida, el $\overline{S}$ debe estar en cero y $\overline{R}$ en uno. Para poner en cero la salida, $\overline{R}$ debe esta en cero y $\overline{S}$ en uno. Si ambas entradas $\overline{S}$ y $\overline{R}$ están en uno, el sistema memoriza (mantiene la salida en su estado anterior). Si ambas estuvieran en cero, se dará el estado prohibido.

Cuando S y R valen "0", la salida Q_f mantiene el valor anterior de Q (Q_i). Cuando R=1 la salida será "0" y cuando S=1 la salida pasa a "1". Esto concuerda con la tabla completa. También queda claro el estado prohibido correspondiente a S y R simultáneamente en 1.

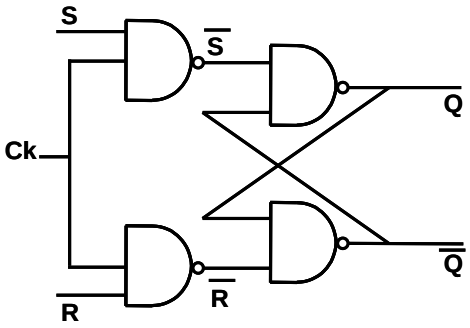
S	R	Q_f
0	0	Q_i
0	1	0
1	0	1
1	1	X

Tabla de verdad reducida de un flip flop R S

En general, podremos reconocer las expresiones algebraicas y explicar el funcionamiento de otros circuitos posibles para un flip flop R S asincrónico.

Los circuitos secuenciales pueden ser sincrónicos o asincrónicos.

Los circuitos sincrónicos necesitan una entrada adicional para sincronizar su funcionamiento. Se la suele llamar "Ck", clock o reloj de sincronización.

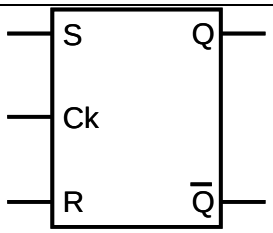


Flip flop RS sincrónico por niveles

Mientras el nivel del reloj sea uno, Ck=1, las entradas "S" y "R" pasan a través de las primeras NAND (ya que el uno lógico es el elemento neutro del producto) y llegan invertidas (negadas) a sus respectivas salidas, **ejerciendo las funciones que hemos analizado anteriormente**.

Cuando el nivel del Ck=0, llega uno (1) a cada entrada $\overline{S}$ y$\overline{R}$, que **corresponde al estado de "memoria"**, en esta versión del flip flop RS, construido con compuertas NAND, independientemente de los valores de las entradas "S" y "R".

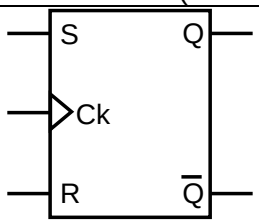
Esta clase de circuito secuencial, es **sincrónico por niveles**. Significa que la sincronización depende del "nivel" en que se encuentre el Ck. El **Ck=0 impide el paso de las señales de entrada y Ck=1, lo permite**. Así el flip flop responderá a las entradas, solo si Ck=1.



Representación del Flip flop RS sincrónico por niveles

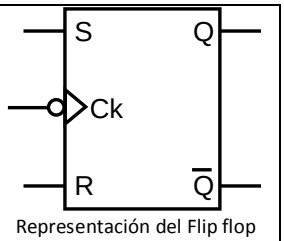
Los flip flop sincrónicos activados por flancos, en cambio, responderán a los valores de las entradas, solamente en la transición (flanco) de la señal aplicada al Ck.

Un flip flop activado por flanco **ascendente**, responderá cuando el Ck pase de cero "0" a uno "1".

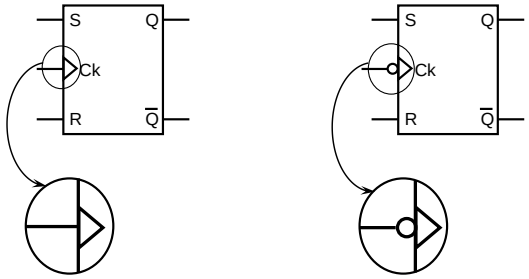


Representación del Flip flop RS sincrónico por flanco ascendente

Un flip flop activado por flanco **descendente**, responderá cuando el Ck pase de uno "1"a cero "0".



Representación del Flip flop RS sincrónico por flanco descendente



Simbología para la representación de la activación por flancos "ascendente" y "descendente", respectivamente

Ejemplo de funcionamiento:

Suponga que el estado actual del flip flop RS es cero "0" en la salida "Q".

Se desea poner en uno la salida.

Para ello, ponemos "1" en "S" y "0" en "R". Sin embargo, estas señales no harán efecto a la salida, hasta que en el Ck se aplique la señal adecuada.

Si se trata de un flip flop RS activado por flancos descendentes (sincrónico activado por flanco descendente), deberá esperar hasta que la señal aplicada al Ck pase de "1" a "0" para obtener el resultado:

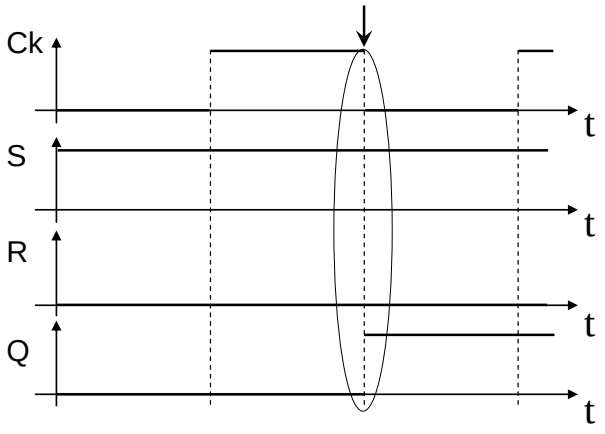


Diagrama de los valores de las señales de entradas y salida, en función del tiempo. Se resalta el efecto del flanco descendente

El nivel bajo de una señal, se asocia (en lógica positiva) a un cero lógico y el nivel alto a un uno lógico.

Para este ejemplo, la salida del flip flop cambia únicamente cuando el Ck pasa de uno a cero (flanco descendente).

Biestables (Flip Flop) JK

Una característica básica del flip flop RS, es el estado prohibido cuando las entradas están simultáneamente en "1" (uno).

Podríamos construir un flip flop que utilice esa condición de las señales de entrada. Por ejemplo que al poner en uno ambas entradas, la salida cambie su estado actual.

De eso se trata el **flip flop JK**.

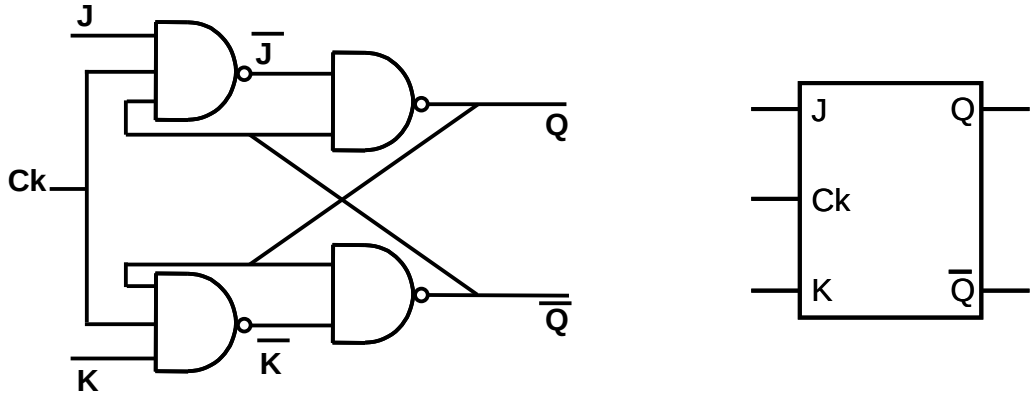
J	K	Q_i	Q_f	
0	0	0	0	Memoria $Q_f = Q_i$
0	0	1	1	
0	1	0	0	Reset $Q_f = 0$
0	1	1	0	
1	0	0	1	Set $Q_f = 1$
1	0	1	1	
1	1	0	1	Invierte el estado inicial: $Q_f = \overline{Q_i}$
1	1	1	0	

Tabla de verdad completa de un Flip Flop JK. Se destacan características de su funcionamiento

Cuando J y K valen "0", la salida Q_f mantiene el valor anterior Q_i. Cuando J=1 la salida pasa a "1" y cuando K=1 la salida será "0". También queda claro que las entradas **J y K simultáneamente en 1, invierte el valor que tenía inicialmente la salida**, ahora $Q_f = \overline{Q_i}$

J	K	Q_f
0	0	Q_i
0	1	0
1	0	1
1	1	$\overline{Q_i}$

Una forma de lograr el funcionamiento descrito:



Flip flop JK sincrónico por niveles

Biestables (Flip Flop) T y su aplicación en contadores

Si unimos las entradas “J” y “K”, J resulta igual a K en todo momento.

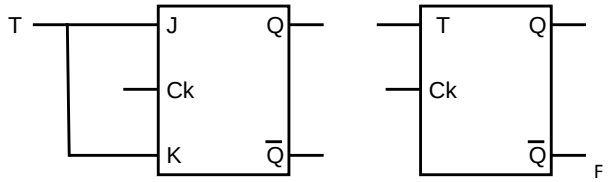
Cuando J=0, K será también 0 y cuando J=1, K lo será también (K=1).

Esto implica que, de las cuatro combinaciones de la tabla de verdad reducida, solo se podrán utilizar la primera y la última combinación.

Este Flip Flop, tendrá ahora una sola entrada efectiva de datos. Se la nombra “T” (la unión de J y K)

J	K	Q_f
0	0	Q_i
0	1	0
1	0	1
1	1	$\overline{Q_i}$

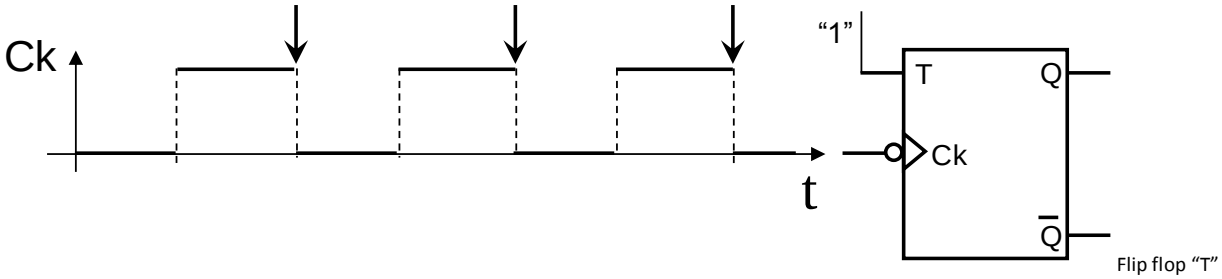
Esto da origen al **Flip Flop T**



Flip flop “T” sincrónico

J=K = T		Q_f
0	0	Q_i
1	1	$\overline{Q_i}$

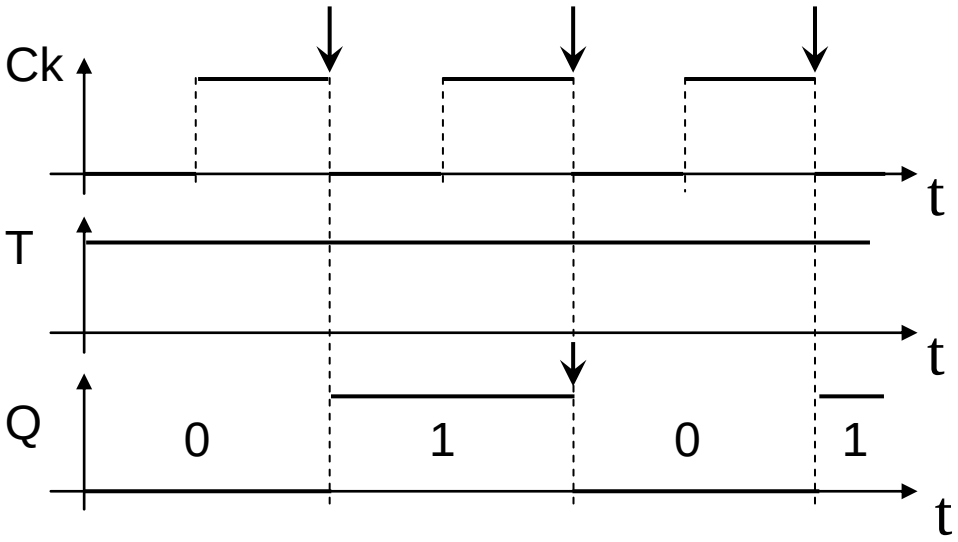
Si en un Flip Flop tipo “T”, mantenemos un uno lógico en su entrada “T”, cada vez que el Ck reciba la señal de habilitación, cambiará la salida “Q”. Supongamos un Flip Flop T, activado por flancos descendentes:



Flip flop “T”

sincrónico por flancos descendentes, excitado por un tren de pulsos en su entrada de Ck

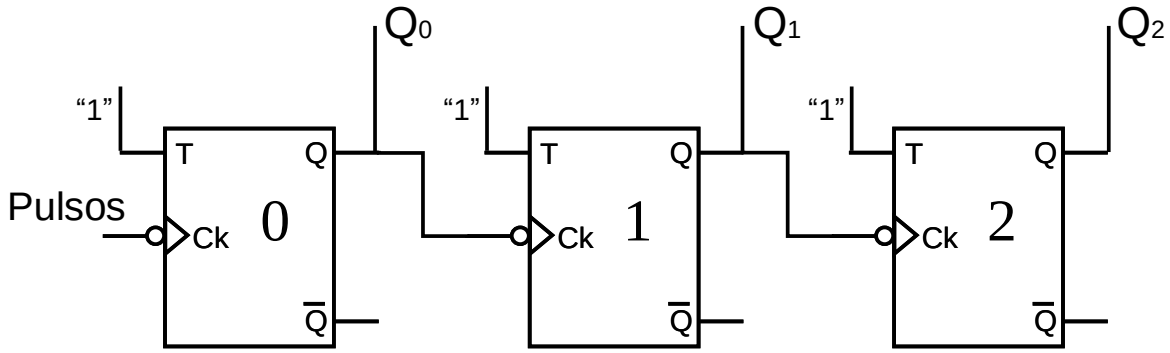
Los distintos valores que adopta la salida “Q” en función del tiempo, se denominan respuesta en el tiempo (respuesta temporal). En este caso, será:



Respuesta temporal de un Flip flop “T” sincrónico por flancos descendentes, excitado por un tren de pulsos en su entrada de Ck

Entre otras aplicaciones, con biestables tipo T pueden construirse **contadores**. Veamos un caso simple.

Contador binario de tres bit.



Contador asincrónico de tres bit, realizado con Flip Flop T de flancos descendentes

Cada Flip Flop, cambiará su estado de salida cuando reciba en su entrada de Ck un flanco descendente.

El Flip Flop "0" cambiará con cada flanco descendente de los pulsos de entrada.
El Flip Flop "1" cambiará con cada flanco descendente de la salida Q_0.
El Flip Flop "2" cambiará con cada flanco descendente de la salida Q_1.

En síntesis, los valores de las salidas Q_2, Q_1 y Q_0, (leídos en ese orden) corresponderán a valores binarios de 000 a 111 (0 a 7 en decimal), para comenzar nuevamente en 0.

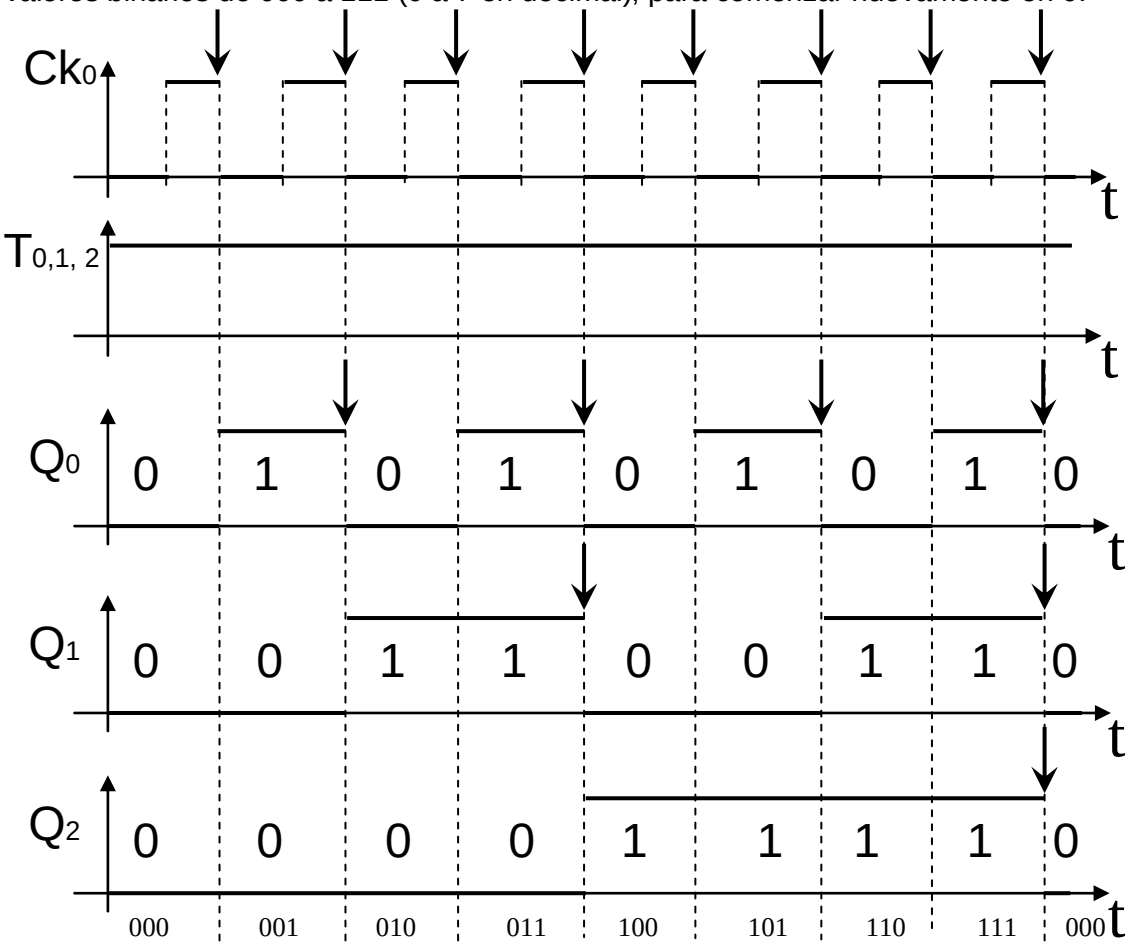
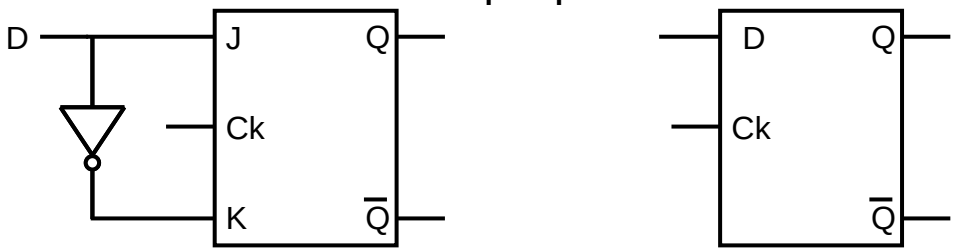


Diagrama de tiempo del contador asincrónico de tres bit, realizado con Flip Flop T de flancos descendentes

Biestables (Flip Flop) D (LATCH) y su aplicación en registros

Si colocamos una compuerta inversora entre las entradas “J” y “K”, K resulta igual al opuesto de J en todo momento. Se trata del **Flip Flop “D”.**



Flip Flop D sincrónico por niveles

Cuando J=0, K será 1 y cuando J=1, K será 0.

Esto implica que, de las cuatro combinaciones de la tabla de verdad reducida, solo se podrán utilizar las dos centrales.

Este Flip Flop, tendrá ahora una sola entrada efectiva de datos. Se la nombra “D” y coincide con el valor de J.

J	K	Q_f
0	0	Q_f
0	1	0
1	0	1
1	1	$\overline{Q}_f$

Siempre y cuando el Ck habilite la entrada del dato, el valor de salida será una copia del valor de la entrada “D”.

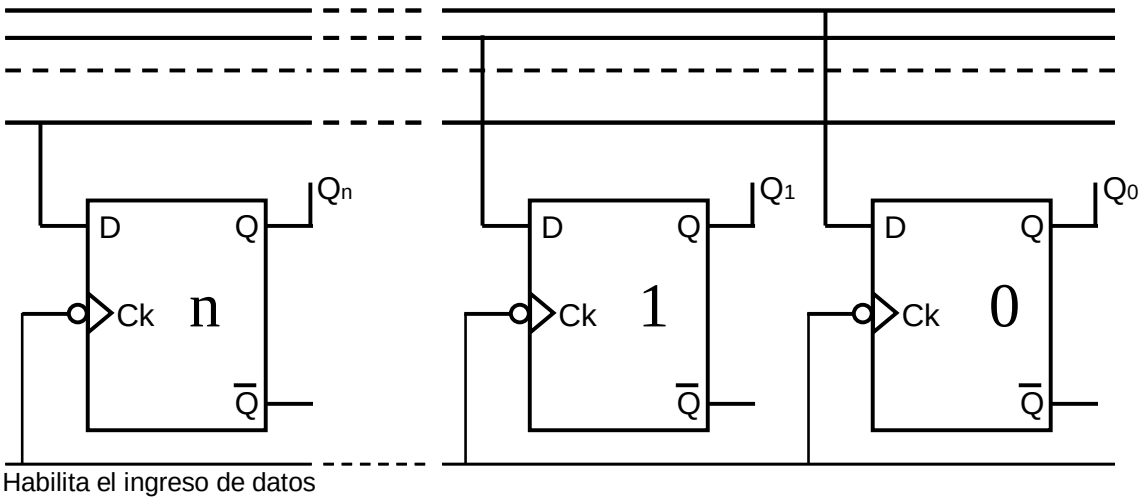
$D = J = \overline{K}$	Q_f
0	0
1	1

Así, si queremos almacenar un bit, bastará con un Flip Flop D.

Registros, registro paralelo.

Si necesitamos almacenar “n” bit, dispondremos “n” biestables “D”, conformando así un registro de “n” bit.

Cuando los Flip Flop reciban la habilitación del Ck, almacenarán y dispondrán a las salidas “Q”, los valores que estaban en las entradas “D” en el momento de la habilitación (sincronización). Si lo hacen al mismo tiempo, simultáneamente, diremos que es un registro paralelo.

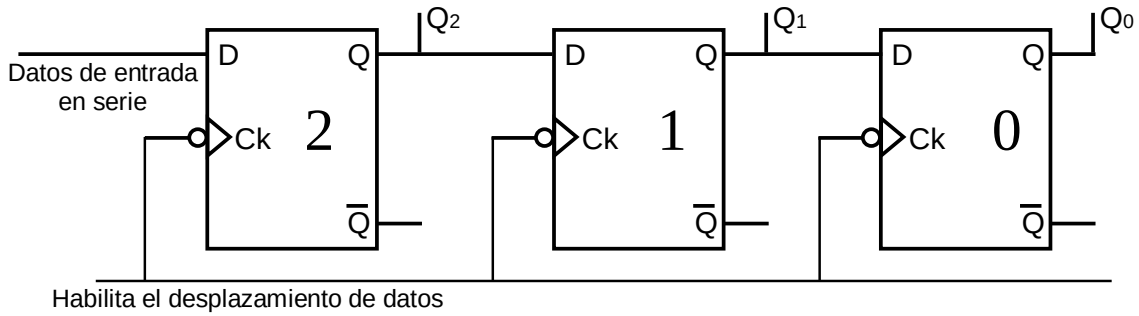


Registro paralelo, a partir de Flip Flop D sincrónicos por flancos descendentes

Registros, registro de desplazamiento (Shift register).

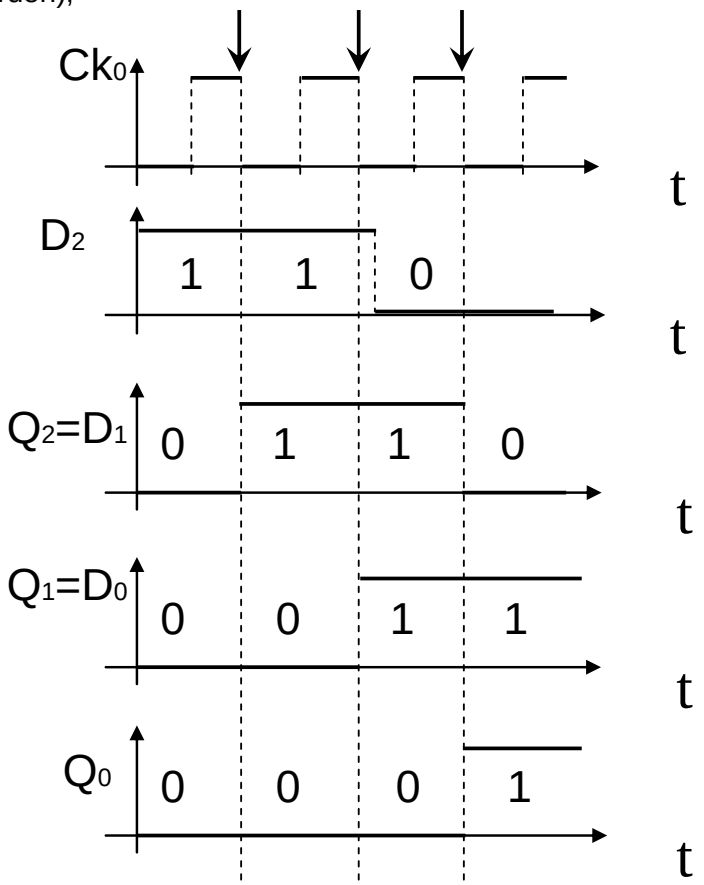
Si conectamos la salida de un Flip Flop D a la entrada de otro, cuando llegue la habilitación del clock (Ck), el dato del primero, se copiará en el segundo.

En el caso de varios Flip Flop acoplados de esta forma, cada pulso de habilitación del Ck, desplazará los datos a la posición del Flip Flop siguiente.



Registro de desplazamiento, a partir de tres Flip Flop D sincrónicos por flancos descendentes

Si los datos ingresados a D_2 de a uno a la vez (Datos de entrada en serie), fueran 0, 1, 1 (en ese orden),



Luego del tercer pulso del Ck, quedan en las salidas Q_2=0, Q_1=1 y Q_0=1.

Es evidente que la información ingresada en serie, ahora queda en paralelo.

Del mismo modo, si un registro de este tipo, tuviera cargado un dato en cada Flip Flop, al recibir los pulsos de Ck igualmente produciría el desplazamiento de datos (en este ejemplo hacia la derecha).

Entonces en la salida Q_0, se irían presentando los valores binarios que estaban almacenados en los otros Flip Flop, de a uno por vez. Es decir, en serie.

Índice

INTELIGENCIA ARTIFICIAL

1. INTRODUCCIÓN

¿Qué es la inteligencia?

Es difícil definir y comprender qué es la inteligencia, aún para aquellos que consagran su vida a estudiarla. Si pidiéramos a varias personas que definan la inteligencia, las respuestas serían diferentes, algunos ejemplos:

- ✓ La capacidad para aprender de la experiencia.
- ✓ La capacidad de discernir, evaluar.
- ✓ La capacidad para razonar.
- ✓ La capacidad para percibir relaciones.

¿Pueden pensar las máquinas?

Esta pregunta es el punto central de la inteligencia artificial.

El objetivo principal de los estudiosos de la computación es desarrollar máquinas que se comuniquen con sus entornos a través de mecanismos sensoriales tradicionalmente humanos y actúen inteligentemente ante situaciones imprevistas sin intervención humana. Esto requiere que la máquina "entienda", o perciba las entradas recibidas y pueda obtener conclusiones gracias a alguna especie de razonamiento. Tanto la percepción como el razonamiento son actividades del sentido común que, si bien son naturales para la mente humana, son bastante difíciles para las máquinas, por lo cuál ésta área está en proceso de desarrollo si consideramos sus metas y expectativas, si bien se han obtenido teorías y técnicas bien fundamentadas.

Examinaremos algunas de ellas, aplicadas en el diseño de máquinas para resolver problemas y que posean capacidades elementales de percepción y razonamiento.

2. Definición de Inteligencia Artificial

La siguiente definición:

> La inteligencia artificial es el estudio de las ideas que permiten a los computadores realizar aquello que hace a las personas parecer inteligentes.
>
> *Patrick Henry Winston, Artificial Intelligence*

captura la idea general de la inteligencia artificial, pero ¿la inteligencia artificial contempla la capacidad de realizar cálculos a velocidades pasmosas?, ¿recordar cientos de direcciones al mismo tiempo?, si alguien pudiera hacerlo, "parecería inteligente". Pero estas acciones no son buenos ejemplos de inteligencia artificial, ya que para un computador son triviales.

Otra definición según *Elaine Rich (Artificial Intelligence)*:

> La inteligencia artificial es el estudio de cómo lograr que los computadores hagan cosas que, por ahora, los seres humanos pueden hacer mejor.

coloca a la Inteligencia Artificial como una "frontera móvil"; en los 50, muchos investigadores de inteligencia artificial trabajaron para crear computadores que pudieran jugar ajedrez, hoy en día pueden derrotar con facilidad a cualquier persona, con excepción de los mejores ajedrecistas, y son pocos los investigadores que estudian este juego.

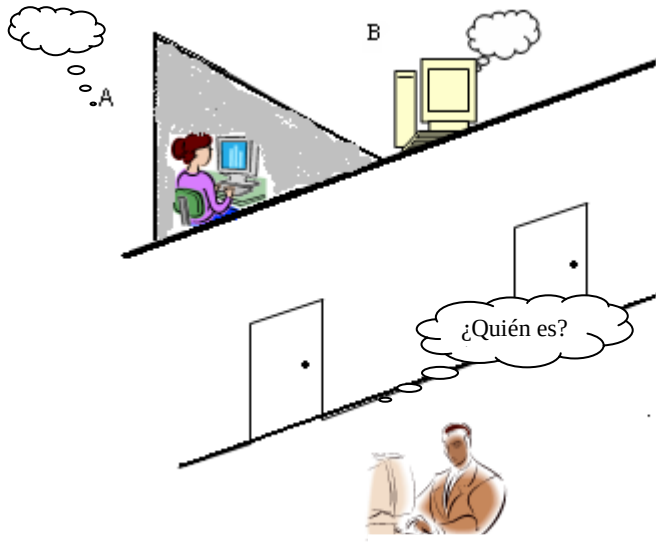
La siguiente definición propuesta por Patrick Winston (*Artificial Intelligence*), resulta ser más concreta y completa:

La inteligencia artificial es el estudio de los cómputos que hacen posible percibir, razonar y actuar.

3. Prueba de Turing

En el fondo, la inteligencia es una característica interior que solo se puede detectar indirectamente desde el exterior, por ejemplo a través de un diálogo estímulo/reacción, que permita diferenciar su existencia real de la simple apariencia de inteligencia.

En los años 50, Alan Turing, teniendo en cuenta la característica engañosa de la inteligencia solo en apariencia, propuso una prueba (Prueba de Turing) para detectar inteligencia dentro de una máquina.



EVALUADOR

Una computadora (B) es INTELIGENTE si un **evaluador humano** no puede distinguir si el que responde a sus preguntas es la computadora (B) u otra persona (A).

Hasta la fecha, ninguna máquina ha logrado pasar la prueba de Turing.

Enfoques de la Inteligencia Artificial

Supongamos que un matemático y un psicólogo emprenden de manera independiente, proyectos para crear un programa que juegue al póquer. Lo más probable es que el matemático diseñe un programa basado en los principios de la probabilidad y la estadística: el resultado seria un programa que jugara sin arriesgarse y no mostraría emociones. El psicólogo, en cambio, probablemente crearía un programa basado en las teorías del raciocinio y el comportamiento humano. El programa del psicólogo podría "involucrarse emocionalmente" en el juego.

Reconsiderando, proponemos la hipótesis de que la preocupación principal del matemático al crear el programa sería su rendimiento final. Se dice que un enfoque así esta orientado al **rendimiento**. En contraste, el psicólogo estaría más interesado en entender los procesos de la inteligencia natural; así, abordaría el proyecto como una oportunidad de probar teorías construyendo modelos de computador basados en esas teorías. Desde este punto de vista, el desarrollo del programa "inteligente" es en realidad un efecto secundario de otro esfuerzo: avanzar en la comprensión del raciocinio y el comportamiento humanos. Se dice que este enfoque esta orientado a la **simulación**.

Ambos enfoques son validos y contribuyen significativamente al campo de la inteligencia artificial.

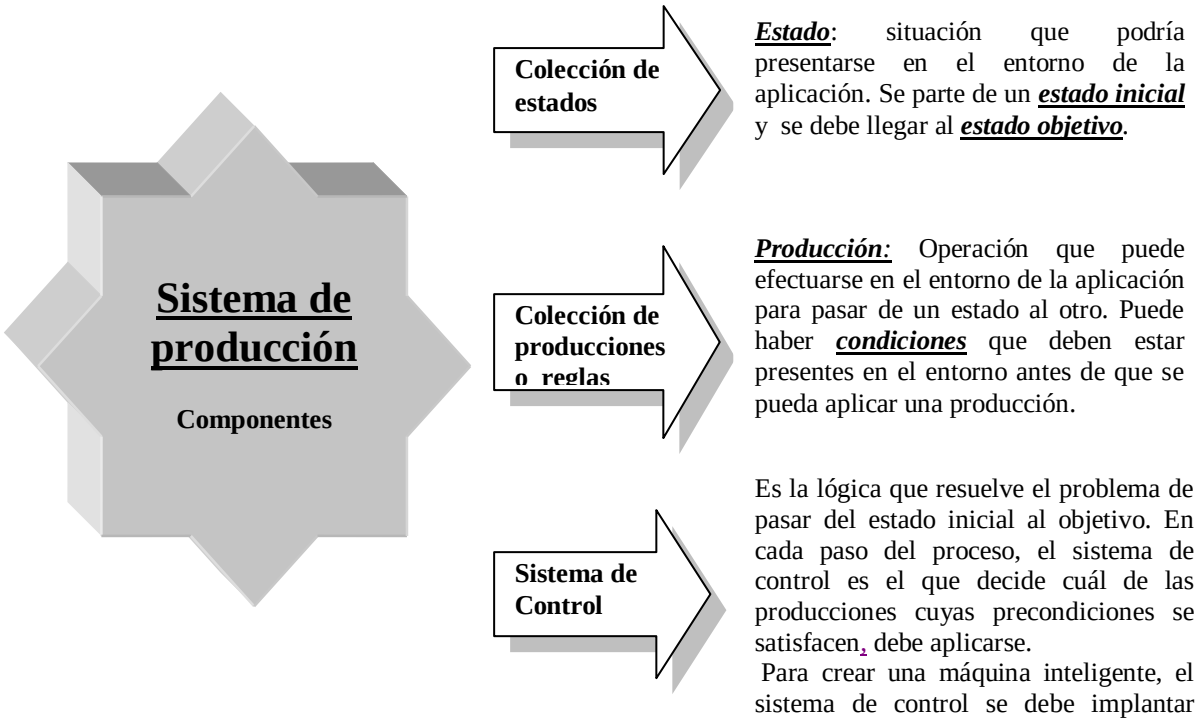
4. Razonamiento

Gran parte del tema de Inteligencia Artificial conlleva un aura de misterio. Con el fin de poner luz al mismo consideremos la demostración de inteligencia a través de un sencillo problema, que No posee un método preestablecido de resolución, donde el programa que se le proporciona a la computadora deberá posibilitar que la misma:

- ✓ Tome decisiones
- ✓ Saque conclusiones
- ✓ Por lo tanto sea capaz de realizar actividades elementales de razonamiento.

El desarrollo de capacidades de razonamiento dentro de una máquina sigue siendo tema de investigación actual, por lo cual existen todavía controversias a cerca de la técnica correcta a seguir.

Aquí abordaremos el tema en el contexto de los **Sistemas de Producción.**



Estado: situación que podría presentarse en el entorno de la aplicación. Se parte de un ***estado inicial*** y se debe llegar al ***estado objetivo***.

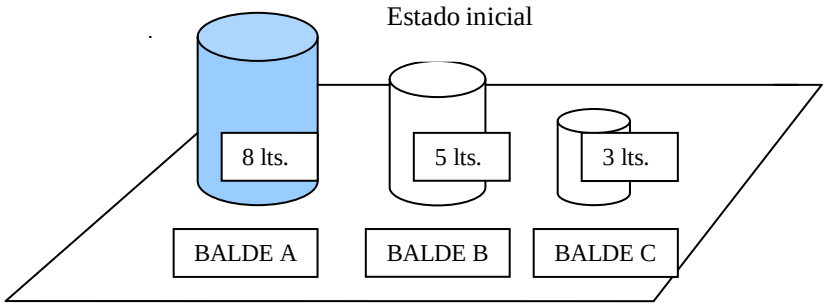
Producción: Operación que puede efectuarse en el entorno de la aplicación para pasar de un estado al otro. Puede haber ***condiciones*** que deben estar presentes en el entorno antes de que se pueda aplicar una producción.

Es la lógica que resuelve el problema de pasar del estado inicial al objetivo. En cada paso del proceso, el sistema de control es el que decide cuál de las producciones cuyas precondiciones se satisfacen, debe aplicarse.

Para crear una máquina inteligente, el sistema de control se debe implantar como un programa almacenado, el cual examina el estado actual del sistema, identifica una secuencia de producciones que conduce al estado objetivo, y ejecuta dicha secuencia.

4.1. Sistema de producción. Problema de los baldes.

Un comerciante tiene tres (3) baldes. El primero de ocho (8) litros de capacidad lleno de líquido, el segundo de cinco (5) litros vacío y el tercero de tres (3) litros también vacío. Los recipientes no están graduados. No se puede agregar ni descartar líquido (en todo momento DEBE haber ocho (8) litros entre los 3 recipientes).



Objetivo:
- Medir cuatro (4) litros en cualquiera de los baldes capaces de contenerlos.
- Los otros cuatro (4) lts. pueden quedar en cualquiera de los otros baldes.

Identifiquemos para este caso los tres componentes principales de un sistema de producción:

1. Colección de estados

Estado inicial	Estado del cual partimos: A= 8 B= 0 C= 0
Estado objetivo	Estado que queremos alcanzar: A= 4 B= ? C= ? o A= ? B= 4 C= ?
Estados válidos	Válido significa: • No volcar agua, en cada paso la suma del líquido de los 3 baldes debe ser siempre igual a 8. • El balde B tiene una capacidad máxima de 5 litros (nunca puede contener más de 5 lts.). • El balde C tiene una capacidad máxima de 3 litros (nunca puede contener más de 3 lts.). • Ningún recipiente puede quedar con menos de 0 lts. (situación absurda para un humano, físicamente imposible, pero posible para la aritmética del procesador) por lo cual deberemos informárselo en las producciones como estado no válido. Es imprescindible informarle a la computadora cuales son los estados válidos y cuales no, de forma que la misma pueda, a través de algún algoritmo, verificar si un estado es válido o no y si es válido buscar la producción más eficaz para llegar desde el estado inicial al objetivo.

2. Colección de producciones o reglas (operación para pasar de un estado a otro):

Producción	Condición previa **SI**	Comentario	Ejecución **ENTONCES**	Comentario
1	**0 < A <= 5 - B**	¿Hay lugar en B para volcar todo A?	**B = B + A** **A = 0**	**Vaciar A en B.** B ahora va a tener su contenido + el de A. A queda vacío.
2	**A > 5 –B > 0**	¿Tiene A agua suficiente para completar B?	**A = A – 5 + B** **B = 5**	**Llenar B con A.** A volcó agua en B hasta completarlo. B quedó lleno

3	**0 < A <= 3 - C**	¿Hay lugar en C para volcar todo A?	**C = C + A** **A = 0**	**Vaciar A en C.** C ahora va a tener su contenido + el de A. A queda vacío.
4	**A > 3 – C > 0**	¿Tiene A agua suficiente para completar C?	**A = A – 3 + C** **C = 3**	**Llenar C con A.** A volcó agua en C hasta completarlo. C quedó lleno.
5	**0 < B <= 8 - A**	¿Hay lugar en A para volcar todo B?	**A = A + B** **B = 0**	**Vaciar B en A. A** ahora va a tener su contenido + el de B. B queda vacío.
6	**B > 8 – A > 0**	¿Tiene B agua suficiente para completar A?	**B = B – 8 + A** **A = 8**	**Llenar A con B.** B volcó agua en A hasta completarlo. A quedó lleno.
7	**0 < B <= 3-C**	¿Hay lugar en C para volcar todo B?	**C = C + B** **B = 0**	**Vaciar B en C.** C ahora va a tener su contenido + el de B. B queda vacío.
8	**B > 3 –C > 0**	¿Tiene B agua suficiente para completar C?	**B = B – 3 + C** **C = 3**	**Llenar C con B.** B volcó agua en C hasta completarlo. C quedó lleno.
9	**0 < C <= 8-A**	¿Hay lugar en A para volcar todo C?	**A = A + C** **C = 0**	**Vaciar C en A.** A ahora va a tener su contenido + el de C. C queda vacío.
10	**C > 8 –A > 0**	¿Tiene C agua suficiente para completar A?	**C = C – 8 + A** **A = 8**	**Llenar A con C.** C volcó agua en A hasta completarlo. A quedó lleno.
11	**0 < C <= 5-B**	¿Hay lugar en B para volcar todo C?	**B = B + C** **C = 0**	**Vaciar C en B.** B ahora va a tener su contenido + el de C. C queda vacío.
12	**C > 5 –B > 0**	¿Tiene C agua suficiente para completar B?	**C = C – 5 + B** **B = 5**	**Llenar B con C.** C volcó agua en B hasta completarlo. B quedó lleno.

3. Sistema de control

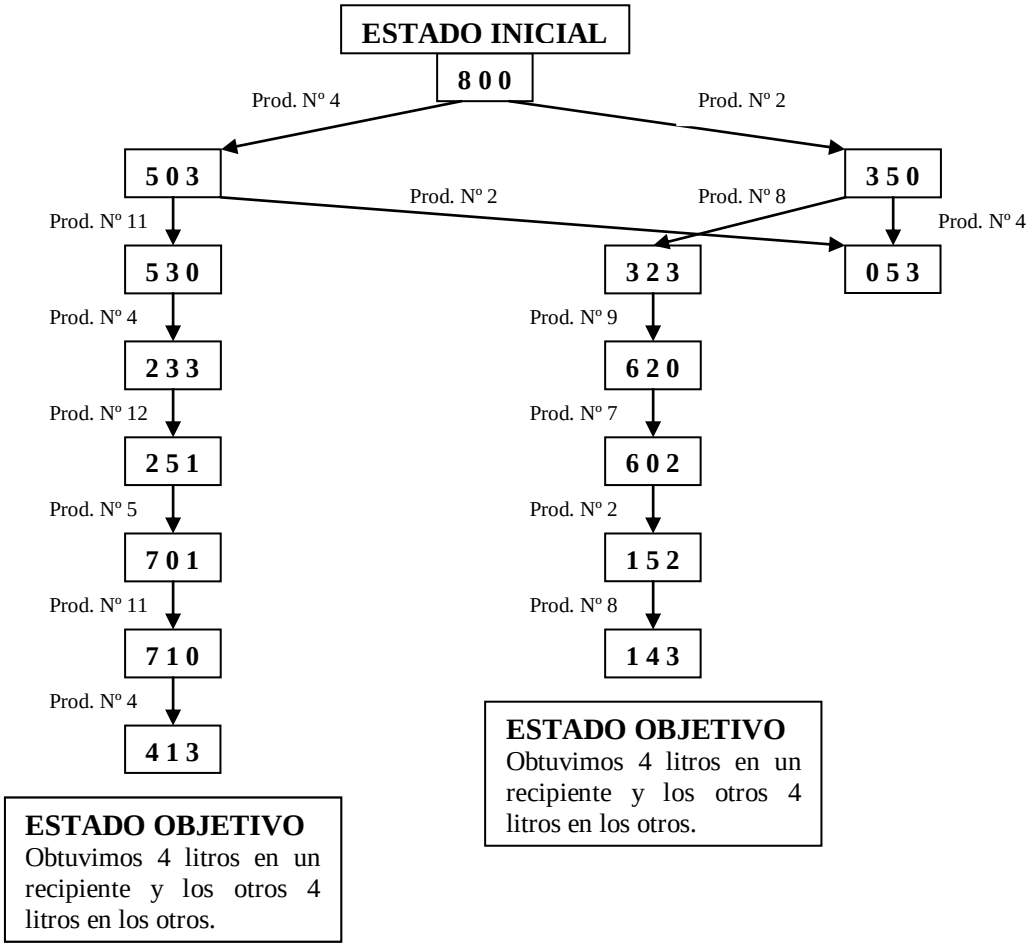
El estado A = 8 B = 0 C = 0, será representado como **8 0 0**

Utilizaremos el concepto de **GRAFO DE ESTADOS,** que es una forma útil de representar, o por lo menos de conceptuar, todos los estados, producciones y condiciones previas de un sistema de producción.

Usamos el término *grafo* en su sentido matemático: una colección de posiciones llamadas *nodos* conectadas mediante flechas llamadas *arcos*.

Grafo de estados: Colección de nodos que representan los estados del sistema conectados por arcos que representan las producciones que causan el movimiento de un estado a otro.

Por lo tanto 2 nodos pueden estar unidos por un arco (en nuestro caso usaremos flechas) en el grafo de estados, sí y solo si existe una producción que puede producir el pasaje de un estado al otro.



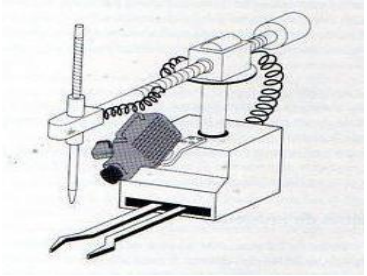
El sistema de control realiza una búsqueda del estado actual de los recipientes en las condiciones previas de cada producción y cuando encuentra una que coincide entonces ejecuta la producción, quedando el estado siguiente como indica esa producción.

Cuando queremos que una computadora, tome decisiones para resolver un problema, debemos "contarle" que significa estado válido y como hace una persona para resolver dicho problema. Mediante instrucciones perfectamente determinadas se escribe un programa que al ser ejecutado por la computadora, le permite llegar a la solución, dotándola de lo que llamamos "Inteligencia Artificial".

5. Una máquina "inteligente"

Con el fin de aclarar varios puntos en cuestión acerca de la "inteligencia" en las máquinas, consideraremos el diseño de una máquina que tiene propiedades elementales de inteligencia.

La misma adopta la forma de una caja de metal equipada con pinzas, una cámara de video y un dedo forrado con caucho para que no resbale al empujar algo.



Está colocada junto a una mesa donde se encuentra un rompecabezas de 8, el cual consiste en un marco cuadrado en donde están montadas ocho fichas, cuadradas y etiquetadas con los números 1 a 8. En el marco caben nueve de estas fichas, por lo tanto hay un hueco hacia el cual puede empujarse cualquiera de las fichas adyacentes a él.

La disposición de las fichas es la siguiente:

1	**2**	**3**
4	**5**	**6**
7	**8**	

Figura 1

Luego, tomamos el rompecabezas y cambiamos la disposición de las fichas empujándolas repetidamente en forma aleatoria hacia el hueco que va quedando, quedando las fichas desordenadas con respecto a su ubicación inicial.

Encendemos la máquina, las pinzas comienzan a abrirse y cerrarse, como pidiendo el rompecabezas. Le entregamos el rompecabezas y las pinzas se cierran sobre él. En unos instantes, el dedo desciende y empuja las fichas dentro del marco (de una manera ordenada), hasta que quedan otra vez en el orden original. Luego la máquina coloca el rompecabezas sobre la mesa y se apaga.

A continuación estudiaremos la capacidad de razonamiento de nuestra máquina.

La acción de abrir y cerrar las pinzas como la de detectar la presencia del rompecabezas, no pueden ser consideradas como acciones inteligentes, ya que el simple mecanismo automático de la puerta de un ascensor es capaz de detectar la presencia de un obstáculo.

5.1. Análisis de imágenes

La primera conducta inteligente que tiene nuestra máquina es la de extraer información a través de la cámara de video. La máquina no se limita a producir y almacenar la imagen del rompecabezas, debe comprender la imagen para poder extraer la posición de los dígitos del rompecabezas, nuestra máquina debe tener la capacidad de *percibir*.

Existen varias técnicas de reconocimiento de imágenes:

- ***Comparación de patrones***: Previamente codificar la imagen del rompecabezas en términos de unos y ceros en la memoria del computador (cada bit representa el nivel de brillantes de una parte específica de la imagen (píxel)) y dando por sentado que la imagen tiene un tamaño fijo, comparar las diferentes secciones de la imagen con los modelos pregrabados (patrones de bits producidos por los dígitos individuales utilizados) al detectarse coincidencias se determina la situación del tablero. Esta técnica utilizada en los lectores ópticos de caracteres (OCR) requiere uniformidad en el estilo, tamaño y orientación de los símbolos que se leen, ya que el patrón de bits producidos por un carácter físicamente grande no coinciden con el patrón de una versión más pequeña. Los problemas se incrementan si el material es manuscrito.

- ***Extracción y evaluación de rasgos***: Comparar las características geométricas más que la apariencia exacta de los símbolos. Así el dígito 1 se podría caracterizar como una sola línea vertical, el 2 una línea curva abierta unida a una recta horizontal y así sucesivamente.
  El proceso consiste en extracción de rasgos de la imagen procesada y posterior evaluación de rasgos (compararlos con los de los símbolos conocidos). No es infalible, errores menores en la imagen, pueden producir rasgos geométricos totalmente distintos, por ejemplo entre un 3 y un 8.

En el problema del rompecabezas de 8 no nos enfrentamos a dificultades mayores como son trabajar con imágenes tridimensionales, comprender imágenes superpuestas, etc. Como vemos los problemas asociados al reconocimiento de imágenes son muchos, pero máquinas pensadas con otra arquitectura (redes neuronales) ayudan a superar algunos obstáculos, más adelante veremos algunos ejemplos.

Una vez descifradas las posiciones de las fichas a partir de la imagen visual, nuestra máquina debe resolver el rompecabezas. Se nos podría ocurrir pre-programar la máquina con soluciones a todas las posibles disposiciones de las fichas, de manera que nuestra máquina solo debería seleccionar y ejecutar el programa apropiado, esto no nos resultaría atractivo ya que aún en un este sencillo rompecabezas puede haber un total de 181.440 configuraciones distintas, pasando a ser poco factibles por limitaciones de tiempo y espacio de almacenamiento en memoria suministrar una solución explícita para cada una.

Por todo esto deberemos programar nuestra máquina para que pueda resolver el problema ella misma, siendo capaz de tomar decisiones, y sacar conclusiones o sea realizar operaciones elementales de razonamiento.

Analizaremos el problema en el contexto de los sistemas de producción, donde el estado inicial es la configuración del rompecabezas cuando se lo entregamos a la máquina, el estado objetivo es la configuración del acertijo resuelto (figura 1).

Las producciones son los movimientos de las fichas, cada movimiento tiene como condición previa que el hueco debe estar junto a la ficha en cuestión.

El sistema de control deberá decidir qué ficha mover de las que se encuentran junto al hueco, o sea qué producción aplicar.

5.2 Árboles de búsqueda

Al estudiar los sistemas de control centraremos nuestra atención en el problema de recorrer grafos, ya que hallar la secuencia apropiada de producciones en un sistema de producción se puede formular en términos de hallar un camino a través de un grafo de estados (o en realidad dentro de lo que llamaremos *árbol de búsqueda*) que lleve del nodo inicial (nodo raíz) al nodo objetivo.

En términos del problema de recorrer grafos, presentamos en primer término una técnica más bien de fuerza bruta, luego veremos como usar la intuición para obtener un sistema más inteligente.

Una técnica común para realizar la búsqueda consiste en recorrer cada uno de los arcos que salen del nodo inicial y registrar el estado al que se llegó, luego recorrer los arcos que salen de esos nuevos estados y una vez más tomar nota de lo obtenido, y así sucesivamente, hasta que uno de los nuevos estados es el objetivo, o sea se encontró una solución. El sistema de control solo debe aplicar las producciones del camino descubierto desde el estado inicial al objetivo.

Se construye así un árbol de búsqueda, partiendo del nodo raíz, donde los hijos de cada nodo son los estados a los que se puede llegar aplicando una producción. Cada arco entre 2 nodos de un árbol de búsqueda representa la aplicación de una sola producción.

En particular, si el rompecabezas de 8 tuviera inicialmente sus fichas posicionadas de la siguiente forma:

1	3	5
4	2	
7	8	6

El árbol de búsqueda resultante sería:

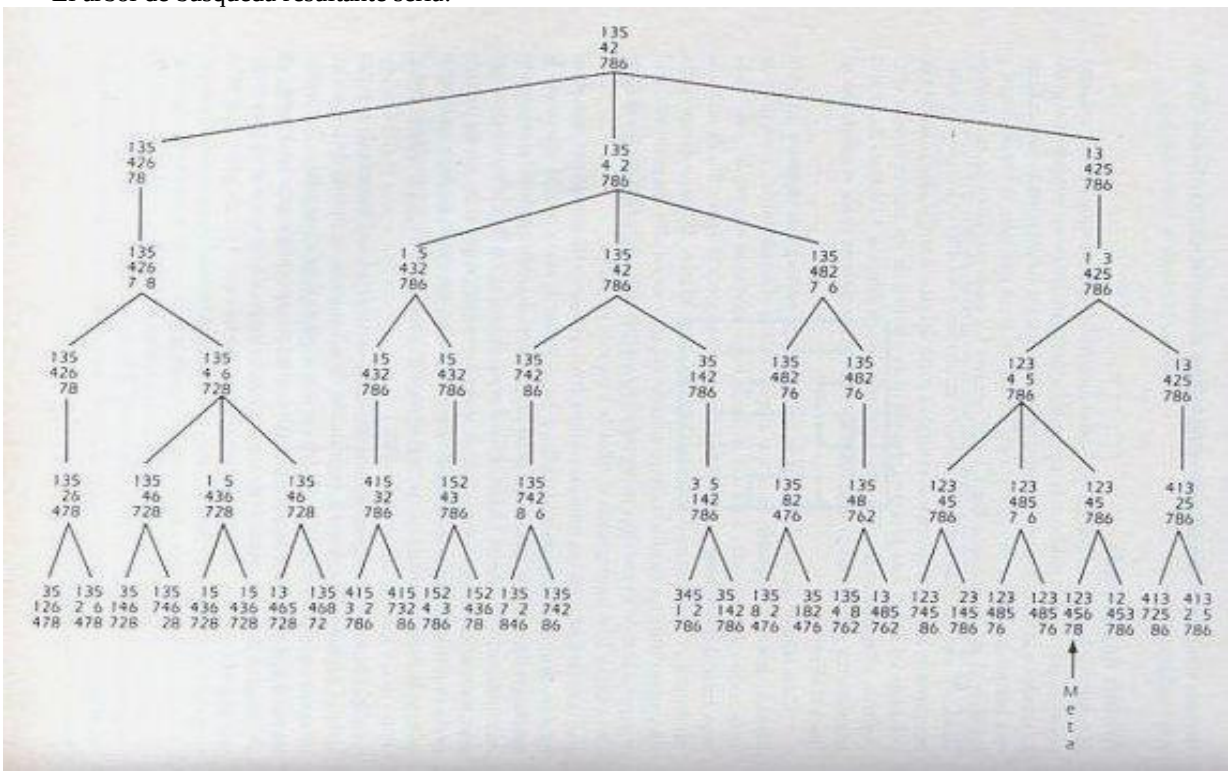


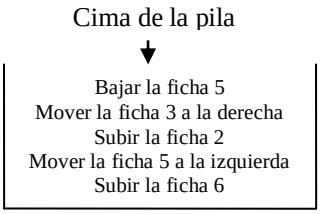
Figura 2

La rama más a la izquierda del árbol representa el intento de buscar una solución moviendo primero la ficha 6 hacia arriba, la central moviendo la ficha 2 a la derecha y la del extremo derecho si comenzamos moviendo la ficha 5 hacia abajo.

Vemos que si comenzamos por subir la ficha 6, la única producción que se permitiría después sería mover la 8 a la derecha, aunque en ese punto también podríamos mover la ficha 6 hacía abajo, lo cual sería inútil pues nos devolvería a un estado ya representado anteriormente.

El estado objetivo aparece en el último nivel del árbol de búsqueda, el sistema de control no necesita construir más niveles del árbol y empieza a subir por el árbol de búsqueda desde el nodo objetivo hasta la raíz (estos árboles se construyen con un sistema de punteros que apuntan hacia arriba en vez de hacia abajo, o con dos series de punteros que permiten desplazarse en ambas direcciones dentro del árbol), guardando en una pila las producciones representadas por cada arco que se encuentra en el camino.

En nuestro ejemplo se obtendría la pila de producciones siguiente:



Esta secuencia de instrucciones servirá para resolver el problema en el mundo externo, o sea la que en realidad va a ejecutar nuestra máquina inteligente.

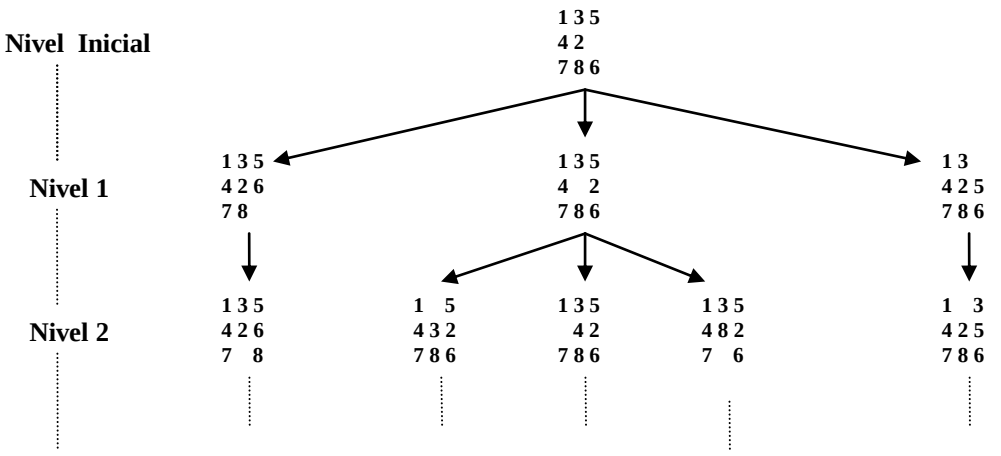
5.2.1 Eficiencia en la construcción de árboles de búsqueda

El ejemplo que hemos elegido produjo un árbol de búsqueda manejable, pero cualquier árbol de búsqueda generado en un intento por solucionar un problema no tan simple como el nuestro, crecería con mayor rapidez que el de nuestro ejemplo, debido al gran número de opciones disponibles en cada etapa y a la mayor profundidad que tendría el mismo antes de hallar la meta.

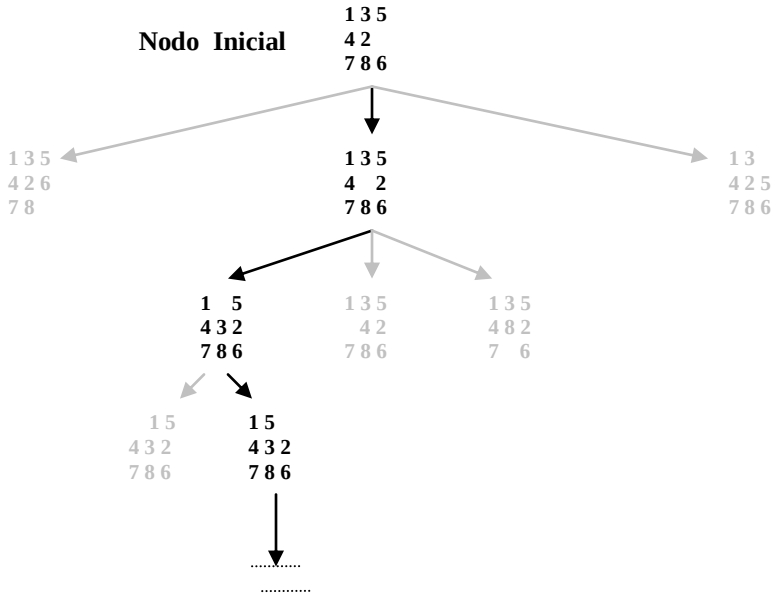
Si pensamos en el juego de ajedrez que tiene 20 primeras jugadas posibles, el nodo raíz tendría 20 hijos en vez de los tres que tiene en el rompecabezas de 8, y un juego de ajedrez fácilmente puede consistir en 30 o 35 pares de jugadas en vez de las 5 sencillas jugadas de nuestro ejemplo. Por todo esto, el desarrollo completo de un árbol de búsqueda resulta poco práctico, como la representación del grafo de estados completo, en término de tiempo y de espacio en memoria. Se necesitan métodos más económicos para manejar el árbol de búsqueda.

- Una forma sería ***cambiar el orden de construcción del árbol.***

En el árbol de la figura 2, la construcción del mismo se realizó ***primero en amplitud*** (nivel por nivel), se fueron creando sucesivas capas horizontales:



En lugar de construirlo primero en amplitud, podemos seguir la ramas más prometedoras (cuál es la más prometedora es tema de la sección siguiente) en profundidad y sólo considerar las otras en caso de no llegar al objetivo. Esta construcción del árbol de búsqueda es ***primero en profundidad,*** en ella se crean caminos verticales en lugar de capas horizontales:



- Otra forma de reducir el tamaño de un árbol de búsqueda es ***evitar redundancias.***

En el caso del rompecabezas de 8 no conectaremos un nuevo nodo al árbol si ese nodo ya está representado anteriormente. En el caso de los recipientes, al llegar a un estado por el cual ya se pasó, se trunca, se abandona esa rama, no se la sigue. Esto resulta demasiado simplista como regla general, en otras aplicaciones puede ocurrir que la nueva ocurrencia del nodo resulte una solución más eficiente que la anterior, y deba añadirse al árbol eliminando la anterior.

Muchos sistemas de control utilizan métodos más complejos para eliminar redundancias en el árbol de búsqueda, generalmente asocian un costo a los diversos caminos representados en el árbol y siguen los que tienen menor costo.

Podemos adoptar este método en el caso del rompecabezas de 8, considerando que el costo de cualquier camino es el número de jugadas que abarca. Si ocurren repeticiones nos quedaremos con el camino menos costoso, o sea el más corto. Si construimos el árbol nivel por nivel, en el caso de repeticiones el nodo retenido será el más viejo, el que aparece más alto en el árbol.

5.3 Empleo de la heurística

Tratando de controlar el tamaño del árbol de búsqueda, veremos que es posible incorporar a nuestro sistema el equivalente de la intuición.

Si nos encontrásemos frente al rompecabezas de 8, lo más probable es que no siguiéramos varias opciones al mismo tiempo, sino que elijamos la opción que nos parece más prometedora y la sigamos, dejándonos llevar por nuestra intuición, aún cuando nos puede conducir al fracaso.

Modificaremos la forma de construir el árbol de búsqueda, incorporando la intuición para que nuestro sistema ahorre tiempo evitando el desarrollo de ramas improductivas en el árbol de búsqueda.

Nosotros, como personas, elegiríamos la opción del estado que se encuentra más cerca del objetivo, en un entorno de programación, debemos desarrollar una medida cuantitativa que permita al programa determinar cuál de varios estados se encuentra más cerca del objetivo. Una medida de estas características se llama heurística.

Información heurística: información empírica no comprobada, que las personas obtienen utilizando su intuición.

Una forma es asociar a cada estado un valor igual al número de fichas que no están en su posición deseada, y considerar el estado con el menor valor como el más cercano a la meta. Aquí no se tiene en cuenta qué tan lejos de su destino están las fichas, por lo tanto podríamos adoptar una medida que tome en cuenta esta distancia.

El proceso consiste para cada estado, sumar el número mínimo de movimientos que debe hacer cada ficha para ubicarse en su posición objetivo (distancia de la ficha), sin tener en cuenta la existencia del resto de las fichas, Por ejemplo una ficha inmediatamente adyacente a su destino final se asocia a una distancia de 1, y aquella cuya esquina toca el cuadrado de su destino final tiene una distancia de 2.

A la suma de las distancias de cada ficha la llamaremos ***Costo Proyectado.*** Por ejemplo, el costo proyectado asociado del siguiente estado es:

2	3	4
5	1	
8	7	6

FICHA	MOVIMIENTOS
1	2
2	1
3	1
4	3
5	1
6	1
7	1
8	1
-------	----------
Costo proyectado	11

Características del costo proyectado

- Nos es útil para tomar una decisión, ya que nos proporciona estimativamente la cantidad de movimientos a efectuar para obtener la solución si llegamos a ese estado.
- Se puede calcular fácilmente, no así el número real de jugadas necesarias para llegar a la meta desde un estado dado.

Ahora que tenemos una heurística la incorporaremos al proceso de toma de decisiones para el rompecabezas de 8. Modificaremos nuestro procedimiento de búsqueda de la Figura 2 considerando el costo proyectado de cada nodo hoja del árbol y continuando la búsqueda en el nodo hoja que posea el menor costo.

El algoritmo para desarrollar un árbol de búsqueda, para un sistema de control que emplee esta heurística es:

- Establecer el nodo inicial del grafo de estados como raíz del árbol de búsqueda y registrar su costo proyectado.
- **MIENTRAS** (no se haya llegado al nodo objetivo) **hacer**

  [Seleccionar el nodo hoja más a la izquierda que tenga el costo proyectado más pequeño de todos los nodos hojas, y conectar como hijos al nodo seleccionado aquellos nodos a los que se puede llegar con una sola producción desde el nodo seleccionado.
  Registrar el costo proyectado de cada uno de estos nuevos nodos junto al nodo en el árbol de búsqueda.]

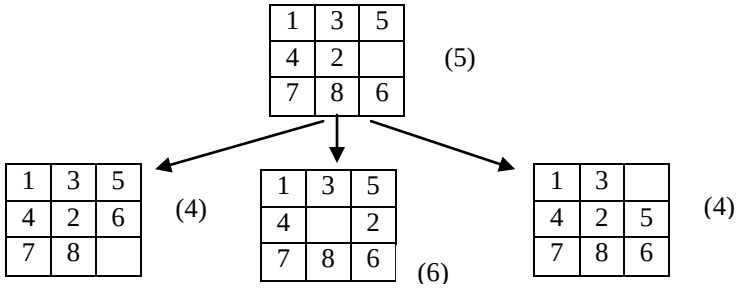
- Recorrer hacia arriba el árbol de búsqueda desde el nodo meta hasta la raíz, metiendo en una pila la producción asociada a cada arco recorrido.
- Resolver el problema original ejecutando las producciones conforme se desempilan.

Recorramos paso a paso este algoritmo aplicado al rompecabezas de 8, partiendo del estado inicial, que será tomada como nodo raíz. Calculamos su costo proyectado, 5.

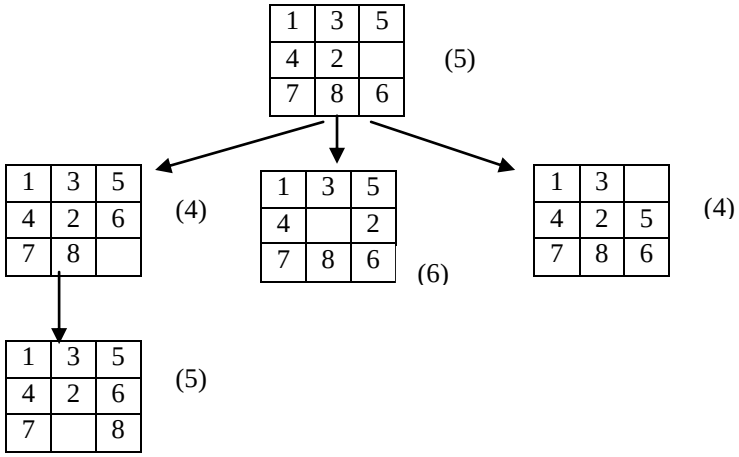
1	3	5
4	2	
7	8	6

(5)

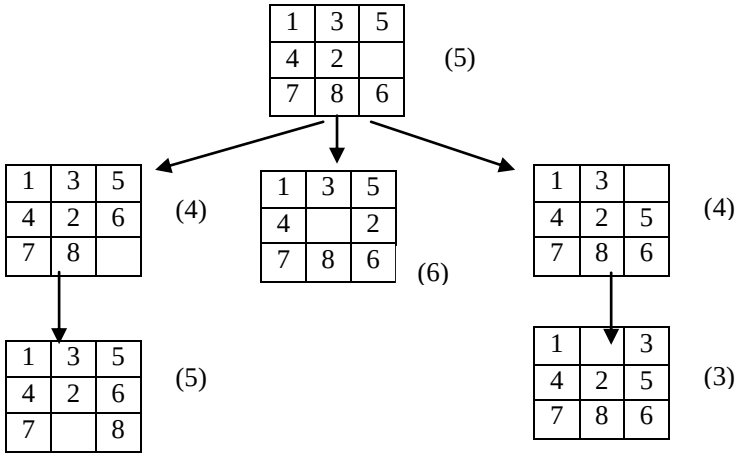
A continuación, la primera ejecución del cuerpo de la estructura ***mientras*** produce el agregado de tres nodos hojas a los cuales se les ha calculado su costo proyectado:



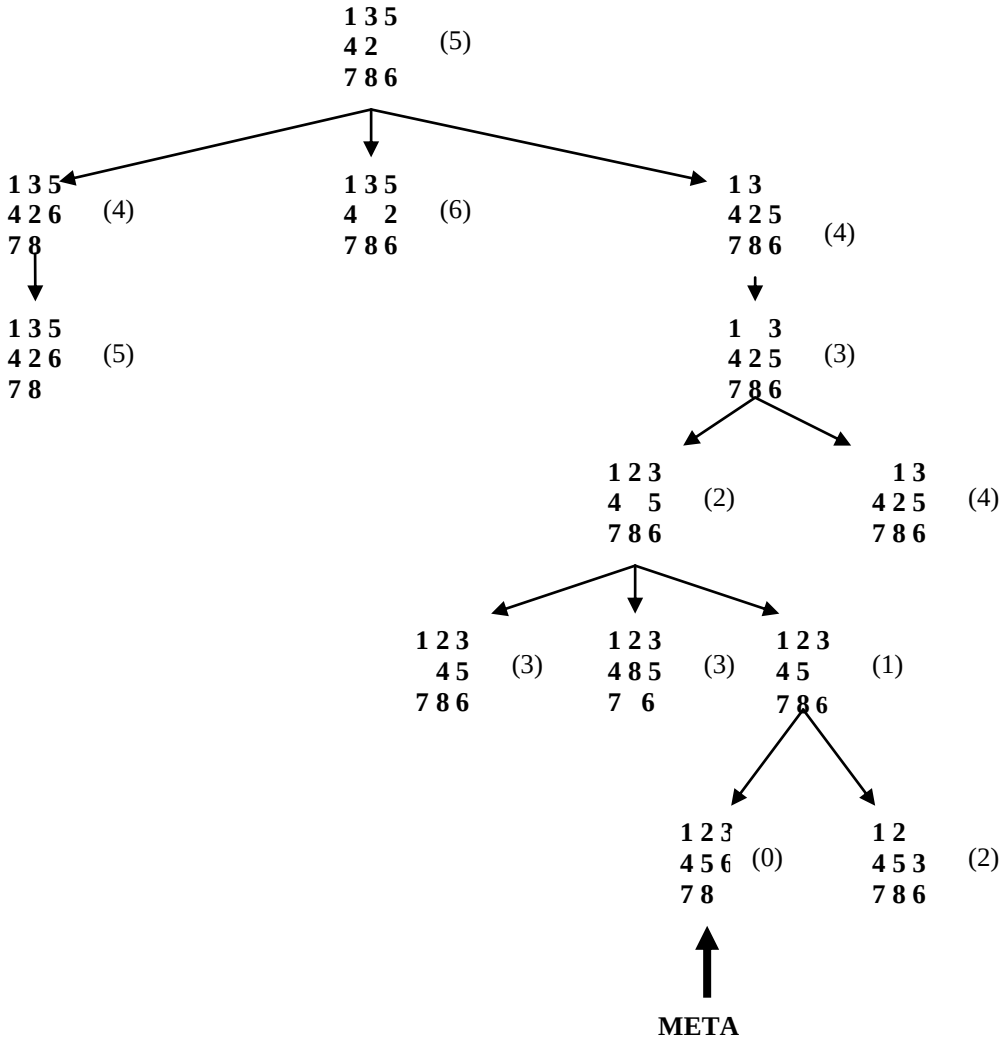
No hemos llegado al nodo objetivo, así que ejecutamos otra vez el cuerpo de la estructura ***mientras***, extendiendo esta vez nuestra búsqueda desde el nodo del extremo izquierdo ("el nodo hoja más a la izquierda que tenga el costo proyectado más pequeño"), luego de lo cuál el árbol de búsqueda se muestra de la siguiente forma:



El costo proyectado del nodo hoja del extremo izquierdo es ahora 5, lo que indica que ésta tal vez no sea una buena opción, el algoritmo descubre esto y en la siguiente ejecución del ciclo ordena expandir el árbol a partir del nodo del extremo derecho, que ahora es "el nodo hoja más a la izquierda que tenga el menor costo proyectado":



Ahora el algoritmo parece ir por buen camino, como el costo proyectado de este último nodo es apenas 3, se ordena seguir por esta rama, por la cual se llega a la meta, y el árbol de búsqueda formado por nuestro sistema de heurística queda de la siguiente forma:

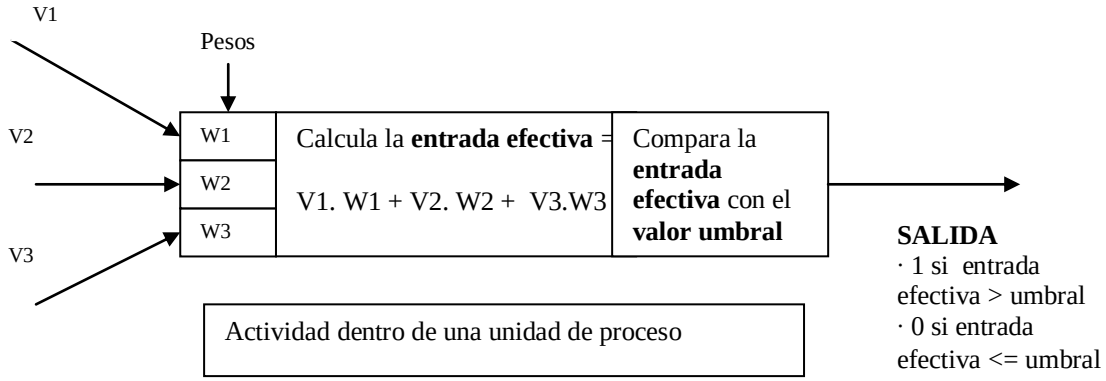


6. Redes neuronales artificiales

Las redes neuronales artificiales es una de las técnicas que utiliza la Inteligencia Artificial para simular el comportamiento inteligente del hombre, ya que poseen un gran potencial para resolver problemas complejos.

Las unidades centrales de proceso que ejecutan secuencias únicas de instrucciones no pueden percibir ni razonar en niveles comparables con los del multiprocesador que es la mente humana, por tal razón se recurre a unidades de procesamiento interconectadas, llamadas neuronas (por su semejanza funcional con las neuronas de los seres humano), las cuales reciben, procesan y transmiten señales, tal cual lo hacen las neuronas en los sistemas biológicos vivos.

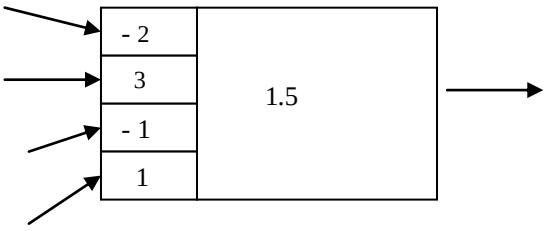
Las *redes neuronales* artificiales están formadas por muchos procesadores individuales, que llamaremos *unidades de proceso*.
Cada unidad de proceso es un dispositivo que produce una salida 1 o 0, según la *entrada efectiva* de la misma exceda un cierto valor umbral.
La *entrada efectiva* es una suma ponderada de las entradas reales.



v1, v2 y v3 son las entradas a esta unidad de proceso (son salidas de otras unidades de proceso), pueden ser 0 o 1 y cada entrada está asociada a un determinado valor llamado *peso* (w1, w2,w3).

Estos pesos pueden ser positivos o negativos según la entrada correspondiente tenga un efecto de excitación o de inhibición sobre la unidad receptora. Si el peso es negativo, un 1 en esa entrada reduce la suma ponderada, tendiendo a mantener la suma ponderada por debajo del umbral, en cambio un peso positivo, aumenta la posibilidad de que la suma exceda el valor umbral. Por lo tanto, ajustando los valores de los pesos en toda la red neuronal, podemos programarla para que responda a diferentes entradas de una manera determinada.

Veamos algunos ejemplos:

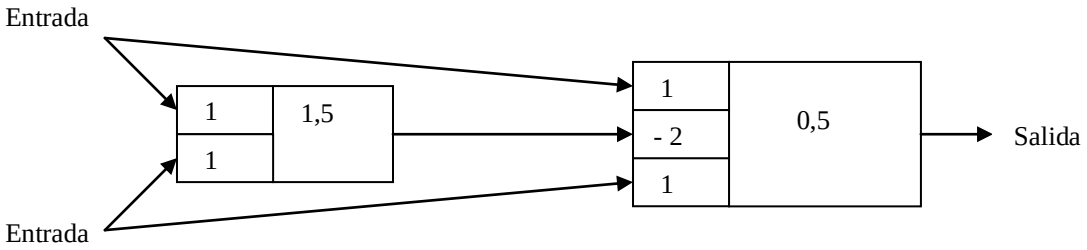


La figura anterior representa una unidad de proceso con cuatro entradas y un valor umbral de 1.5. Las entradas tienen los pesos siguientes: la primera se pondera con el valor -2, la segunda con 3, la tercera con -1 y la cuarta con 1.

Si la unidad recibe las entradas 1,1,1 y 0, su entrada efectiva será (1)(-2) + (1)(3) + (1)(-1) + (0)(1) = 0, y por lo tanto su salida será 0.

En cambio si la unidad recibe 0,1,1 y 1, su entrada efectiva será (0)(-2) + (1)(3) + (1)(-1) + (1)(1) = 3, que supera el umbral y por lo tanto su salida será 1.

La siguiente red neuronal, está programada para producir una salida de 1 si sus dos entradas son distintas y una salida de 0 en caso contrario:



7. Aplicaciones de la Inteligencia Artificial

Consideraremos algunas de las áreas en las cuales la Inteligencia Artificial tiene aplicación.

7.1 Procesamiento de lenguajes naturales

El procesamiento de lenguajes naturales (PLN) se ocupa de la formulación e investigación de mecanismos eficaces computacionalmente (que se puedan realizar por medio de programas) para la comunicación entre personas o entre personas y máquinas por medio de lenguajes naturales.

Una de las aplicaciones es en el terreno de la Traducción Automática.

Los sistemas tradicionales están diseñados de modo de buscar el enunciado original (o una parte del mismo) en una tabla en la cual está almacenado junto con su equivalente traducido, en estos casos la máquina no tiene que entender enunciados sólo reconocer su sintaxis. Pero la traducción de lenguajes naturales (idiomas) como el inglés, español, etc. requiere una "comprensión " del enunciado para ser correcta. Por ejemplo, para traducir los enunciados

María rompió con su novio.

Y

Maria rompió esos platos.

se requiere que el traductor sea capaz de comprender los mismos. En esta tarea el principal inconveniente es que las personas pocas veces se ajustan a reglas estrictas cuando hablan y generalmente no dicen lo que quieren decir. El proceso de entender un enunciado en un lenguaje natural requiere varios niveles de análisis:

Análisis sintáctico	Identifica el papel gramatical de cada palabra**.** Ejemplo: María lavó la camisa de Pedro. Se identifica a María como sujeto del enunciado
Análisis semántico	Busca identificar la acción descripta, el agente de esa acción y el objeto de la acción. Ejemplo: reconocería que "María lavó la camisa de Pedro" y "La camisa de Pedro fue lavada por María" dicen lo mismo.

Análisis contextual	Se considera el contexto del enunciado en el proceso de comprensión. Ejemplo: el enunciado " El tiempo cambió de repente", solo podemos saber su significado si conocemos su contexto.

7.2 Reconocimiento automático de la voz.

Los digitalizadores de audio son dispositivos de entrada que pueden capturar palabras habladas, música y otros sonidos para poder almacenarlos como datos digitales. Pero deben ser procesados por software complejo antes que el computador pueda interpretarlos como palabras.

Los sistemas de reconocimiento automático de voz usan técnicas de reconocimiento de patrones similares a las que se emplean en los sistemas de visión y de reconocimiento óptico de caracteres (OCR):

- Segmentación de los patrones de sonido de entrada para formar palabras y fonemas individuales.
- Reglas expertas para interpretar los sonidos.
- "Expertos" de contexto para manejar sonidos ambiguos.
- Aprendizaje de un entrenador humano.

Existen grandes diferencias entre las voces humanas, por lo cual es necesario entrenar muchos de los sistemas comerciales actuales para que reconozcan la voz de una persona en particular. Aún así, trabajaran de manera confiable si el usuario habla de manera pausada y usa un vocabulario pequeño, previamente definido. Las investigaciones actuales apuntan a superar estas limitaciones y producir sistemas que tengan como características:

- Independencia del hablante.
- La capacidad de manejar la voz sin límite de vocabulario.
- La capacidad de manejar voz continua, el habla natural a velocidad normal.

Hasta ahora no existe sistema, salvo el cuerpo humano, que logre estos tres objetivos.

A pesar de las limitaciones, los sistemas de reconocimiento de voz son usados por trabajadores de fábricas y otras personas que tienen las manos ocupadas mientras que usan el computador.

Es posible comunicar números y órdenes por teléfono para operaciones bancarias automatizadas, verificación de tarjetas de crédito y otras aplicaciones remotas. Reducen las limitaciones de muchos minusválidos pues les permiten dar órdenes verbales a computadores y dispositivos robóticos.

7.3 Robótica

En ningún lugar es más visible la tecnología de la Inteligencia Artificial que en el campo de la robótica, la visión, la audición, el reconocimiento de patrones, la toma de decisiones, la comprensión del lenguaje natural, el habla, están presentes en los robots actuales.

La diferencia más importante de hardware entre los robots y otros computadores son los periféricos de entrada y salida. En vez de enviar la salida a una pantalla o a la impresora, un robot envía órdenes a articulaciones, brazos y otras partes móviles.

La mayoría de los robots cuentan con algún tipo de sensores de entrada, que les permite corregir o modificar sus acciones con base en la retroalimentación del mundo exterior.

Un robot o máquina a la cual se le pide tomar determinadas piezas y colocarlas en una caja, las piezas llegan por una cinta a intervalos regulares y las cajas llenas son reemplazadas por cajas vacías en forma consistente en el mismo lugar, o sea la máquina realiza su tarea en un entorno controlado, una aplicación de este tipo no incluye inteligencia.

Surge una diferencia importante cuando la máquina debe realizar sus tares en un entorno no controlado, por ejemplo en la exploración espacial. En nuestro ejemplo supongamos que las piezas montadas llegan en una caja con otros tipos de piezas, en vez de estar montadas en la cinta transportadora la tareas de la máquina consiste en reconocer la pieza correcta, dejar de lado las otras piezas, y tomar los objetos buscados.

Si suponemos que las piezas se colocan en las cajas en forma arbitraria, la recuperación de las piezas requiere una secuencia única de pasos que se debe desarrollar dentro de la máquina misma, la cual deberá vigilar y entender constantemente la situación ya que es posible que las piezas dentro de la máquina se muevan, necesitando modificarse las actividades requeridas. Todas estas cuestiones caen dentro del ámbito de la Inteligencia Artificial.

Los robots son utilizados en fábricas, para desactivar bombas, soldar tubos en el fondo del mar, realizar trabajos en centrales nucleares, cirugías automatizadas para implantes de cadera, etc.

En el caso de los robots espaciales, estos sustituyen al hombre en tareas que son demasiado peligrosas, difíciles, repetitivas o incluso imposibles para los astronautas, teniendo que cumplir algunas exigencias específicas:

- Resistir un lanzamiento.
- Funcionar en condiciones ambientales difíciles, algunos ejemplos:
  - la baja presión en la órbita provoca que el frío suelde las partes metálicas entre si.
  - la radiación es diferente de la encontrada en la Tierra, en el espacio, las partículas pesadas hacen que la electrónica digital se comporte mal.
- Funcionar autónomamente. Ya que son manejados lejos de su base, las señales para controlarlos y supervisarlos tiene que viajar durante mucho tiempo produciendo retrasos en las comunicaciones que impiden la tele-operación en tiempo real, por lo tanto, deben ser capaces de funcionar solos y solucionar cualquier problema que ocurra mientras realiza sus tareas.

7.4 **Sistemas de bases de datos**

Los sistemas de almacenamiento y recuperación de datos representan una aplicación importante de los sistemas de proceso de los lenguajes naturales. El objetivo es poder solicitar información a estos sistemas usando un lenguaje natural en vez de ajustarnos a un lenguaje técnico. También se utilizan técnicas de Inteligencia Artificial en el proceso de responder al usuario.

Sistemas tradicionales	**Sistemas de inteligencia artificial**
- Recuperan solo los datos solicitados explícitamente	-Recuperan también información que no fue solicitada explícitamente. Por ejemplo: en las búsquedas legales, si se quiere tener información de casos similares se acostumbra pedir al usuario que identifique una palabra que oriente la búsqueda, si la palabra fuera "menores" un sistema inteligente traería también información de aquellos casos que contiene la palabra "infantes".
- Recuperan sólo los datos almacenados previamente	-Recuperan también información que no fue directamente solicitada pero está relacionada. Por ejemplo: si se tiene una base de datos con información acerca de los presidentes argentinos y se desea preguntar si algún presidente argentino midió más de 3 metros, el sistema tradicional no podría dar respuesta a menos que se guardara la altura de todos los presidentes. Un sistema inteligente, podría responder (NO) sin tener las alturas, ya que razona: si hubiera habido algún presidente de tres metros de altura, al ser un dato significativo estaría guardado en la base de datos.
- Se limitan a contestar literalmente la pregunta hecha.	- Trata de averiguar qué desea saber realmente el usuario del sistema o qué debe decírsele. Ejemplo: solicitamos información acerca de la cantidad de alumnos "promocionados " por el profesor Pérez, nos contesta "cero" podríamos decir que dicho profesor es bastante exigente, luego preguntamos por los "aplazados" y por los que cursaron la materia, en ambos casos la respuesta es "0", ante la situación sospechosa preguntamos si Pérez dictó la materia , la base de datos contesta "no". Un sistema inteligente debería habernos contestado eso al principio.

7.4.1 Sistemas expertos

Sistemas expertos: Paquetes de software diseñados para ayudar a las personas en situaciones en las que se requiere un experto en un área específica

El desarrollo de sistemas expertos es una extensión importante del concepto de bases de datos inteligentes. Se construyen de modo que simulen el razonamiento de causa y efecto que seguirían los expertos en dicha situación. Así, por ejemplo, un sistema experto médico propondrá el mismo procedimiento que un experto médico humano que sabe que debe realizarse una biopsia si se observa alguna anormalidad y si una radiografía indica la presencia de una masa en ese lugar.

Para construir un Sistema Experto se deberá:

- Obtener los conocimientos requeridos de un experto. Esto representa dos cuestiones:
  - Lograr y mantener la cooperación del experto, esto es difícil ya que en general las consultas suelen ser largas y el experto es reacio a compartir sus conocimientos con un sistema que podría sustituirlo.
  - La mayoría de los expertos desconocen el proceso de razonamiento que siguen para sacar conclusiones.

- Organizar los conocimientos obtenidos del experto en un formato compatible con un sistema de software, en general se expresan los conocimientos como una colección de reglas en forma de enunciados **SI-ENTONCES,** por ejemplo la regla de nuestro sistema experto médico se puede expresar como:

  SI se percibe anormalidad
  y la radiografía indica presencia de masa
  ENTONCES realizar biopsia.

Existe una similitud entre las reglas de un sistema experto y las producciones de un sistema de producción. La parte "si" de la regla expresa las condiciones para efectuar el enunciado de la parte "entonces".

De hecho, muchos sistemas expertos son en lo esencial sistemas de producción en los que las reglas obtenidas del experto son las producciones y el sistema de control simula el razonamiento basado en esas reglas.

Sistema de producción	Se llama	Sistema Experto
Colección de producciones	→	Base de conocimientos del sistema
Sistema de Control	→	Máquina de inferencia
Se le encomienda llegar a una meta predeterminada	→	No necesariamente se le encomienda llegar a una meta predeterminada, más bien se le pide dar recomendaciones fundamentadas

El motor de inferencia es la estructura de control de un sistema experto, contiene el programa que gestiona la Base de Conocimientos y otros mecanismos necesarios para administrar un sistema de naturaleza interactiva.

El sistema experto se puede dividir en dos componentes, unos de razonamiento y el otro de conocimientos. Gracias a esto, es factible aplicar el sistema de rutinas de razonamiento a distintas situaciones, añadiendo una nueva base de conocimientos, obteniendo nuevos sistemas expertos. De la misma manera que un sistema de control puede aplicarse a distintos problemas con solo sustituir las producciones por las que corresponden a dichos problemas