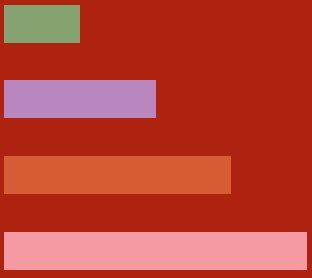
Matemáticas para Todo

MateCompu1

lógica y circuitos

Versión: 3 de julio de 2023 16:21:41 -06:00

Manuel López Mateos



MLM
EDITOR

2022

Manuel López Mateos

Matemáticas para Todo

MateCompu1

lógica y circuitos

Versión: 3 de julio de 2023 16:21:41 -06:00

Manuel López Mateos

MLM
EDITOR

2022

1. Conjuntos, lógica y funciones.
   https://clf.mi-libro.club/

2. MateCompu - lógica y circuitos.
   https://matecompu.mi-libro.club/

Edición corregida, 2022

Versión: 3 de julio de 2023 16:21:41 -06:00
https://matecompu.mi-libro.club/

Información para catalogación bibliográfica:
López Mateos, Manuel.
MateCompu1 - lógica y circuitos / Manuel López Mateos — 1a ed.
x–107 p. cm.

1. Matemáticas 2. Ciencias de la computación 3. Programación 4. Sistemas numéricos 5. Bits 6. Bytes 7. Computadoras 8. Electrónica digital 9. Mecatrónica I. López Mateos, Manuel, 1945– II. Título.

Producido en México

Prefacio

Esta obra forma parte de una colección que con el pomposo título de *Matemáticas para Todo* pretende exponer los elementos de varios temas usados en cada vez más amplias y diversas disciplinas.

Aunque cada vez más jóvenes tienen acceso a equipos de cómputo, en su mayoría lo utilizan para diversión, soslayando la gran utilidad de las computadoras para comprender y transformar nuestro *habitat*, tanto natural como social.

El nivel escolar del bachillerato es ideal para interesar a jóvenes en el estudio de las ciencias de la computación y en el uso inteligente de sus equipos de cómputo.

Muchos de los temas de cómputo al alcance de quien estudie bachillerato emplean matemáticas que desafortunadamente no se enseñan en ese nivel.

Ese es el propósito de este libro: presentar varios temas de matemáticas accesibles, para el estudio del *cómputo* y del *arte de programar* y que no forman parte de los planes y programas de estudio.

En este breve volumen iniciamos con una introducción a los temas de conjuntos, lógica y circuitos, subrayando la similitud entre las operaciones en cada tema.

A lo largo de la exposición veremos temas diversos de ciencias de la computación y del arte de programar.

Entre los temas mezclamos *Problemas* con los cuáles se pretende impulsar a quienes estudien esta obra a realizar su propio

descubrimiento al resolverlos. Recomendamos que traten de resolver los problemas, aunque al final del libro está la solución de cada uno.

Agradezco los comentarios de Román Alberto Velasquillo García y José Galaviz, y la cacería de erratas de Nicole Bragaña.

Pueden plantear o resolver dudas en el grupo MateCompu, de Facebook.

Advertencia Aunque el material presentado es indispensable para posteriores cursos de capacitación o especialización en manejo de dispositivos eléctricos, electrónicos, digitales o de diversa tecnología, esta obra **NO** constituye una Guía, Instructivo ni Manual de operaciones. Las **Actividades** sugeridas a lo largo del libro, relacionadas con manejo de material eléctrico o de cualquier otro tipo, **deberán realizarse en laboratorios o talleres, bajo la *supervisión* de personal capacitado**. El manejo equivocado de material eléctrico puede ocasionar accidentes que causen daño físico, iniciar incendios, pérdidas o daños a instalaciones e incluso pérdida de vidas humanas.

Manuel López Mateos
manuel@cedmat.net
3 de julio de 2023
16:21

aportación voluntaria

Capítulo 1

Inicio

1.1. ¿Prendido o apagado?

Son los dos estados de un foco. Un foco puede estar *prendido* o *apagado*. Una proposición lógica puede ser *verdadera* o *falsa*. Estos *estados excluyentes* los representamos con los números `0` (apagado) y `1` (prendido). Si una proposición lógica es verdadera, le asignamos el *valor de verdad* `1`, de no ser así, si es falsa, su *valor de verdad* es `0`.

`0`, foco apagado `1`, foco prendido

Figura 1.1 Usamos los números `0` y `1` para representar dos estados excluyentes.

En una computadora un *bit*[1] es una componente electrónica que, a semejanza de un foco, tiene dos estados excluyentes posibles: prendido o apagado.

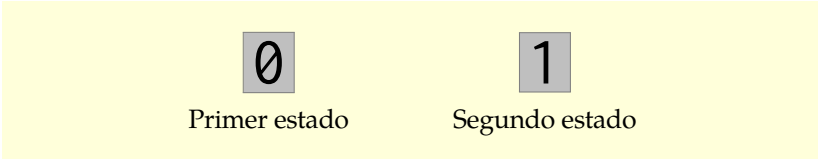
[1] La palabra *bit* es una combinación de ***Bi**nary digi**t*** (dígito binario).

Se trata de la unidad básica de información usada en cómputo, un *dígito binario* sólo puede tener dos valores, que se representan con `0` y `1`; se considera un dispositivo con dos estados.

> Un *bit* es un dispositivo que tiene dos estados excluyentes que se representan con `0` y `1`.

El *primer* estado del dispositivo es `0`, equivalente a apagado.
El *segundo* estado del dispositivo es `1`, equivale a prendido.

> En cómputo se comienza a *contar* desde el `0` que es el primer estado de un *bit*.



Además de comenzar a contar desde `0`, numeramos los bits de un dispositivo, de *derecha* a *izquierda*.

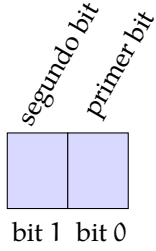


Figura 1.2 Dispositivo de dos bits.

¿Cuántos estados tiene un dispositivo de dos bits? Sabemos que cada bit tiene *dos* estados: `0` y `1`.

Primer estado: 0 0 ambos bits apagados

Segundo estado: 0 1 primer bit (el bit 0) prendido

Tercer estado: 1 0 el segundo bit (el bit 1) prendido

Cuarto estado: 1 1 ambos bits prendidos

Para cada estado del bit 0 hay dos estados posibles del bit 1; como hay dos estados posibles del bit 0, tenemos que en total hay $2 \times 2 = 2^2 = 4$, es decir cuatro estados posibles en el dispositivo de dos bits.

En el caso de tres bits, para cada uno de los dos estados del bit 2 tenemos cuatro estados posibles para los dos primeros bits (el bit 0 y el bit 1), es decir hay $2 \times 2^2 = 2^3 = 8$ estados posibles en un dispositivo de tres bits.

	bit 2	bit 1	bit 0
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
7	1	1	0
8	1	1	1

Figura 1.3 Al aumentar un bit se duplica el número de estados.

Cada bit tiene dos estados, luego un dispositivo de cuatro bits tiene $2^4 = 16$ estados y así sucesivamente, un dispositivo de n bits tiene 2^n estados.

Se usó un grupo de 7 bits para codificar el conjunto de caracteres y acciones conocidas como código ASCII[2].

A los bits necesarios para codificar un carácter en una computadora se le llamó *byte*[3]. El tamaño de un byte ha cambiado según la arquitectura de las máquinas.

> Actualmente, un *byte* es un grupo de 8 *bits*, usado para codificar información, ya sean caracteres o números.

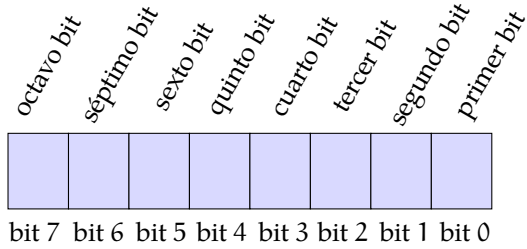


Figura 1.4 Comenzamos contando a partir de 0.

Cada bit tiene dos estados posibles, luego un byte tiene $2^8 = 256$ estados posibles. Por ejemplo

| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | ó | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | .

La manera de contar los estados de un dispositivo de varios bits es un caso particular del

[2] Se pronuncia *asqui*, es el acrónimo de *American Standard Code for Information Interchange* (Código estadounidense convencional para intercambio de información).

[3] La palabra *bit* en inglés significa *pedacito*. En junio de 1956, Werner Buchholz empleó la palabra *bite*, que significa *mordida*, para denotar grupos de bits; le cambió la ortografía y surgió el vocablo *byte* (se pronuncia *bait*), Buchholz, «The Word *Byte* Comes of Age...»

Principio general del conteo. Si es posible realizar una tarea de m maneras y otra de n maneras, las dos tareas se pueden realizar de $m \times n$ maneras.

Ejemplo 1.1 Puedes escoger *pasta* o *ensalada*. De aderezo puedes escoger *aceite de oliva*, *queso parmesano* o *aceitunas*. ¿Cuántos platillos tienes para escoger?

Solución. La base la puedes escoger de 2 maneras (ensalada o pasta) y el aderezo lo puedes escoger de 3 maneras (aceite de oliva, queso parmesano o aceitunas), luego, por el principio fundamental del conteo, hay $2 \times 3 = 6$ maneras de escoger un platillo. ☺

Problema 1.1 Hay un casco de vikingo, una peluca con largas trenzas y un sombrero con máscara de *El Zorro*; un *kilt* escocés, un mameluco, un *tutú* y una bata de hospital; unos huaraches y unas botas de montar. ¿Cuántos atuendos (de tres prendas) se pueden formar?

1.2. ¿Está o no está?

Los temas de *Conjuntos* y *Lógica* facilitan la comunicación y ayudan a organizar ideas. Aquí daremos una breve introducción, para más detalles les recomiendo mi obra *Conjuntos, lógica y funciones*.

El concepto de *conjunto* es uno que no se puede definir empleando el lenguaje cotidiano. Hagan una prueba, pregunten a varias personas

¿Qué es un conjunto?

y analicen las respuestas.

De seguro que algunas responderán «es una reunión de objetos», otras dirán «es una agrupación de objetos», unas más «es un montón de objetos», y cosas por el estilo. En el intento de definir el concepto, simplemente se refieren a un sinónimo; si insistimos y preguntamos ahora «¿Qué es una *agrupación*?» veremos que responderán con algún otro sinónimo.

No es posible *definir el concepto* de Conjunto con lenguaje cotidiano, pero podemos usarlo.

Si hablamos del conjunto de los estados de un dispositivo de un bit, sabemos de qué se está hablando, de los dígitos `0` y `1`. Sabemos que la letra A no es un estado de un bit, ni que la palabra *chancla* lo es.

Aunque hay ejemplos complicados, entendemos lo que se quiere decir cuando alguien se refiere a *un conjunto de*….

Para que un conjunto sea tal, le pedimos que esté *bien definido*, lo cual significa que dado un objeto y un conjunto, podamos decidir si el objeto pertenece o no al conjunto[4].

Ejemplo 1.2 El conjunto de los estados de un bit está bien definido. Los elementos de ese conjunto son `0` y `1`. Cualquier otro objeto ***no*** está en el conjunto. ☺

Si un objeto está en un conjunto, ese objeto ***es*** un ***elemento*** del conjunto. Esa pertenencia la denotamos con el símbolo $\in$. Si llamamos B al conjunto de estados de un bit, entonces

$$\texttt{0} \in \mathrm{B}, \quad \texttt{1} \in \mathrm{B}, \quad \mathrm{A} \notin \mathrm{B}, \quad \textit{chancla} \notin \mathrm{B},$$

donde el símbolo $\notin$ significa que ese objeto ***no es*** un elemento del conjunto.

Un conjunto se puede *listar* o *describir*.

[4] Para no caer en la Paradoja del barbero. *Ver* Russell, *Paradoja de Russell — Wikipedia, The Free Encyclopedia* y López Mateos, *Conjuntos, lógica y funciones*, p. 2.

Si B es el conjunto de estados de un bit lo listamos como

$$B = \{ 0, 1 \},$$

es decir, entre llaves colocamos cada uno de sus elementos.

O lo podemos describir

$$B = \{ x \mid x \text{ es un estado del bit} \},$$

que se lee «B es el conjunto de x tales que (la raya vertical $|$ se lee *tales que* o *tal que*) x es un estado del bit».

Aunque no siempre lo mencionamos de manera explícita, cuando hablamos de elementos y de conjuntos, estos elementos son objetos que pertenecen a un conjunto más general, que llamamos ***total*** o ***conjunto universo*** y denotamos con Ω, *Omega* mayúscula, la última letra del alfabeto griego.

Para referirnos a un conjunto C de camisas azules, consideramos que Ω es el conjunto de camisas y entonces

$$C = \{ x \in \Omega \mid x \text{ es azul} \}$$

que se lee «C es el conjunto de camisas x tales que x es azul».

Si tenemos un conjunto A de elementos de Ω y resulta que algún objeto $z \in \Omega$ no pertenece a A, entonces pertenece a su ***complemento***, es decir, el complemento de un conjunto A, lo denotamos con A^c, con $\overline{A}$ o con A', es el conjunto de elementos de Ω que no están en A,

$$A' = \overline{A} = A^c = \{ x \in \Omega \mid x \notin A \}.$$

Lo ilustramos con el **diagrama intuitivo**[5] de la Figura 1.5.

[5] En multitud de libros de texto, a estos **diagramas intuitivos** se les llama, de manera incorrecta, *Diagramas de Venn*. Puedes ver la explicación en López Mateos, *Conjuntos, lógica y funciones*, sección 4.3, página 103.

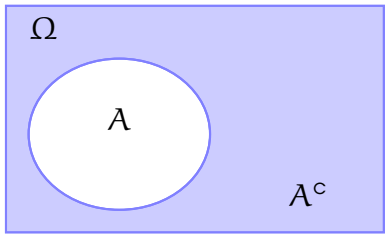


Figura 1.5 La parte sombreada es el complemento de A.

Ejemplo 1.3 Sea Ω el conjunto de estados de un dispositivo de dos bits. ¿Cuál es el conjunto A de elementos de Ω que tienen prendido el bit `0`? ¿Cuál es el complemento de A?

Solución. Conjunto de estados de un dispositivo de dos bits:

$$\Omega = \{\, \texttt{00}, \texttt{01}, \texttt{10}, \texttt{11} \,\};$$

de ellos, los estados con el bit 0 prendido forman el conjunto

$$A = \{\, \texttt{01}, \texttt{11} \,\}.$$

El complemento de A será el conjunto de estados que ***no*** tengan el bit 0 prendido,

$$A^c = \{\, \texttt{00}, \texttt{10} \,\}.$$ ☺

Si sucede que dos conjuntos A y B son tales que cada elemento de A es un elemento de B, decimos que A es un ***subconjunto*** de B o que A está ***contenido*** en B; lo denotamos con $\subseteq$,

Ejemplo 1.4 Si X es el conjunto de múltiplos positivos de 4, menores que 10 y Y es el conjunto de múltiplos positivos de 2, menores que 10, ¿está X contenido en Y?

Solución. Listemos cada conjunto,

$$X = \{4, 8\}$$
$$Y = \{2, 4, 6, 8\},$$

vemos que *sí*, cada elemento de X es un elemento de Y. ☺

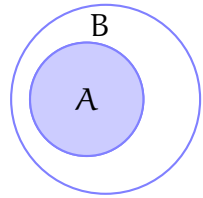


Figura 1.6 $A \subseteq B$ si cada elemento $x \in A$, está en B.

La ***contención*** de conjuntos es ***transitiva***, es decir,

$$\text{si } A \subseteq B \text{ y } B \subseteq C \text{ entonces } A \subseteq C.$$

Para verificar que dos conjuntos A y B son ***iguales***, debemos verificar que

1. $A \subseteq B$ y que
2. $B \subseteq A$,

es decir, la ***doble contención***.

Problema 1.2 Sea Ω en conjunto de estados de un dispositivo de cuatro bits. Demuestra que los estados que tienen prendido el bit `0`, `2` y `3`, están contenidos en el conjunto de estados que tienen prendido el bit `0` y `3`.

Hay un conjunto peculiar, el ***conjunto vacío***, es el conjunto que ***no*** tiene elementos, lo denotamos por $\emptyset$,

$$\emptyset = \{ x \in \Omega \mid x \neq x \}.$$

El conjunto vacío es útil para indicar que algo no sucede. Supongamos que tenemos un dispositivo de tres bits y Ω es el conjunto de los estados. ¿Cuál es el conjunto F de los estados que tiene algún bit que no esté prendido ni apagado? Claramente ningún bit cumple esas características, luego $F = \emptyset$.

Problema 1.3 Sea Ω el conjunto de platillos posibles según el Ejemplo 1.1 de la página 5. ¿Cuál es el conjunto Q de platillos que llevan queso parmesano? ¿Cuál es el complemento de Q? ¿Cuál es el conjunto R de platillos que llevan *salsa catsup*?

1.3. ¿Verdadero o Falso?

Una ***proposición lógica*** o simplemente una ***proposición***, es una afirmación que tiene *dos* posibles ***valores de verdad***, V ó F, es verdadera o es falsa.

Dada una proposición p, es *verdadera*, V, o es *falsa*, F.

Si una proposición p es verdadera, su *negación*, que denotamos con $\neg p$, es falsa. Lo describimos en la ***tabla de verdad*** de la negación,

$\mathbf{p}$	$\mathbf{\neg p}$
V	F
F	V

En la columna $\mathbf{p}$ vemos los posibles valores de verdad de p, que son verdadero V, o falso F. En la columna $\mathbf{\neg p}$ vemos los *correspondientes* valores de $\neg$p. Cuando p tiene valor V, $\neg$p tiene valor F. Cuando p tiene valor F, $\neg$p tiene valor V.

Ejemplo 1.5 Di cuáles de las siguientes expresiones son proposiciones, da su valor de verdad y enuncia su negación.

p: 2 es un estado de un bit,

q: Un Estudio en Escarlata,

r: 32 es mayor que 7,

s: ¡Ni tú / ni yo!

Solución. p ***sí*** *es una proposición*, es Falsa (los estados de un bit son `0` y `1`), su negación es

$$\neg p: \quad 2 \text{ no es un estado de un bit,}$$

la cual es Verdadera.

q ***no*** *es una proposición*, es el título de un libro[6], no afirma o niega nada y no tiene un valor de verdad, y por lo tanto no tiene negación.

r ***sí*** *es una proposición*, es Verdadera pues es cierto que el número 32 es más grande que 7. Su negación es

$$\neg r: \quad 32 \text{ no es mayor que } 7,$$

la cual es Falsa.

s ***no*** *es una proposición*. Se trata de los últimos dos renglones de un poema[7]. Sin saber a qué se refiere, no podemos saber su valor de verdad. ☺

Problema 1.4 Enuncia la negación de cada proposición y di su *valor de verdad*.

1. El número 5 es par.
2. La Tierra gira alrededor del Sol.
3. Hay hongos venenosos.

[6] Se trata de la novela que da inicio la saga de Sherlock Holmes. *Ver* Conan Doyle, *A Study in Scarlet*.

[7] *Primera Cohetería*, García Lorca, *Obras Completas*, Tomo I, , [p. 801].

1.4. Proposición, *bit* y conexión

Nota la semejanza entre *proposición* y *bit*. Una proposición es *Verdadera* o *Falsa*. Un bit está *prendido* o *apagado*.

Si una proposición es *Verdadera*, su negación es *Falsa*, es decir, podemos pensar a la *negación* como una ***operación*** que cambia el *valor de verdad* de una proposición.

$$\begin{aligned} \text{p Verdadera} &\to \neg\text{p Falsa} \\ \text{p Falsa} &\to \neg\text{p Verdadera} \end{aligned}$$

De manera análoga, la ***negación***, aplicada a ***un bit***, que se denota con $\sim$, cambia su *estado*.

Si x es el estado de un bit y aplicamos la *negación*, lo prendido se apaga y lo apagado se prende:

$$\begin{aligned} x \text{ Prendido} &\to \sim x \text{ Apagado}, \\ x \text{ Apagado} &\to \sim x \text{ Prendido}. \end{aligned}$$

Lo cual abreviamos como

$$\begin{aligned} \text{Si } x = 0 \text{ entonces } \sim x &= 1, \\ \text{Si } x = 1 \text{ entonces } \sim x &= 0. \end{aligned}$$

O simplemente, aplicado a bits,

$$\sim 0 = 1, \quad \sim 1 = 0.$$

En un ***circuito eléctrico***, una ***conexión*** tiene dos estados: *abierta* (no pasa la corriente) y *cerrada* (sí pasa la corriente).

Figura 1.7 Conexión *abierta* y conexión *cerrada*.

Las conexiones también se conocen como ***interruptores***.

1.5. Negación y *bytes*

La operación de *negación* aplicada a bits, también se aplica a dispositivos de varios bits, si denotamos con x un estado del dispositivo, denotaremos con $\sim x$ o con `NOT` x, al estado del dispositivo obtenido al cambiar el estado de cada bit por su "negativo", es decir al intercambiar los *ceros por unos* **y** los *unos por ceros*.

Ejemplo 1.6 Si $a = \texttt{0110}$ es un estado de un dispositivo de *cuatro* bits, entonces $\sim a = \texttt{1001}$, o simplemente,

$$\sim\texttt{0110} = \texttt{1001}.$$

Para ilustrar que la negación intercambia `0`s y `1`s, colocamos el resultado de aplicar `NOT` como

$$\begin{aligned}&\texttt{NOT 0110}\\ =&\texttt{1001}\end{aligned}$$ ☺

Ejemplo 1.7 Usualmente se representa un *byte*, que es un dispositivo de *ocho* bits, como dos grupos ligeramente separados de *cuatro* dígitos binarios, para facilitar su lectura; es decir, en lugar de `01001011`, escribimos `0100 1011`. Tenemos entonces que

$$\sim\texttt{0100 1011} = \texttt{1011 0100}.$$

O, ilustrando el intercambio de `0` y `1`,

$$\begin{aligned}&\texttt{NOT 0100 1011}\\ =&\texttt{1011 0100}.\end{aligned}$$ ☺

La *negación* se considera como una operación análoga al *complemento* de un conjunto, de hecho, se usa la expresión "*complementar los bits de un* byte" para indicar el intercambio de *ceros* y *unos* del byte.

Ejemplo 1.8 Si *complementamos* los bits de `0000 0100` obtenemos `1111 1011`.

Es decir,

$$\sim\texttt{0000 0100} = \texttt{1111 1011}$$

O, ilustrando los intercambios

$$\begin{aligned}&\texttt{NOT 0000 0100}\\ =&\texttt{1111 1011}.\end{aligned}$$

Problema 1.5 Considera un dispositivo de cuatro bits

1. ¿Cuáles son los estados de un dispositivo de cuatro bits? Sea Ω dicho conjunto.
2. Sea A el subconjunto de los estados que tienen prendido el primer y tercer bit (es decir el bit 0 y el bit 2). Halla A^c.
3. Para cada elemento de A halla su *negativo*, forma con ellos el conjunto B. ¿Es cierto que $A^c = B$?

Capítulo 2

Operaciones lógicas

El complemento de un conjunto, la negación de una proposición o del estado de un bit y el cambio del estado de una conexión (interruptor), son operaciones efectuadas sobre **un** objeto. Es una operación *unaria*. Cuando una operación se aplica a dos objetos, se llama ***operación binaria***, como la suma o el producto de números.

Como en el caso del *complemento*, también para las operaciones binarias básicas de *intersección* y *unión* en conjuntos, tenemos operaciones similares en proposiciones, bits y conexiones.

Un ***circuito eléctrico*** es una disposición cerrada de cables conductores, conexiones o interruptores y dispositivos, unida a un *potencial* (comúnmente llamado *fuente*), representado con $\vert\vert$; por donde se desplazan *cargas eléctricas*.

En 1886, el científico y filósofo estadounidense Charles Sanders Peirce, en una carta a su ex-alumno Allan Marquand, mencionaba que la electricidad podía ser la base para construir una máquina que pudiera resolver problemas matemáticos muy difíciles (Ver la Figura 2.12 en la página 29).

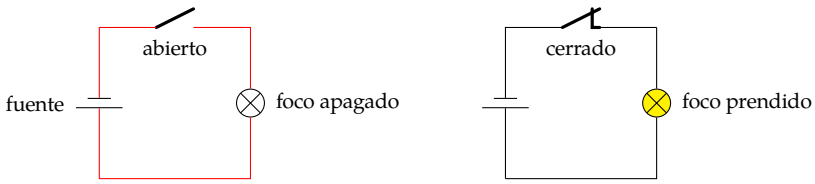


Figura 2.1 Fuente, cables conductores, interruptor, foco.

Con un diagrama o circuito ***lógico*** representamos el estado de un dispositivo D a partir del estado de uno o varios interruptores de un circuito eléctrico. Es decir, en un ***circuito lógico*** *circula* ***información*** que se modifica por medio de ***compuertas lógicas***.

Un interruptor sólo puede estar *abierto* (`0`) o *cerrado* (`1`), así, la información que circula en un circuito lógico son `0`s y `1`s.

La negación en un bit actúa como un ***interruptor*** o *switch*, pues cambia su estado.

2.1. Complemento, negación, `NOT`

Recordemos, el ***complemento*** de un conjunto A está formado por los elementos del *universo* Ω que no pertenecen a A; son los *puntos de* Ω que no están en A,

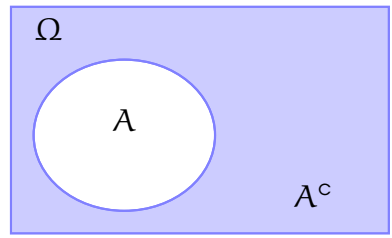


Figura 2.2 $A' = \overline{A} = A^c = \{\, x \in \Omega \mid x \notin A \,\}$.

Una ***proposición*** p tiene dos valores de verdad, es *verdadera* (V) o *falsa* (F). La negación ¬p tiene los valores de verdad opuestos a p, su tabla de verdad aparece del lado izquierdo de la siguiente figura: cuando p es verdadera, ¬p es falsa, y viceversa.

La negación se comporta de manera análoga en un bit y en un interruptor.

La tabla del lado derecho muestra que si el estado x de un bit es `0`, ***apagado*** (el interruptor está ***abierto***), la negación ~x es `1`, ***prendido*** (interruptor ***cerrado***).

p	¬p
V	F
F	V

x	~x
0	1
1	0

Compuerta lógica NOT

El dispositivo lógico que cambia un estado es la ***compuerta lógica*** NOT, la representamos con el símbolo  .

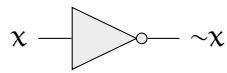


Figura 2.3 La ***compuerta lógica*** NOT invierte el estado.

Usemos la compuerta lógica NOT para construir un dispositivo físico cuyo estado sea el negativo de la entrada.

x	~x = D
0	1
1	0

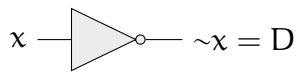


Figura 2.4 Tabla de verdad, diagrama lógico.

Supongamos que D es un foco y la compuerta NOT es un interruptor. La figura anterior dice que el estado del foco es el

negativo de la entrada x. Así, si $x = 0$, es decir si x está abierto, entonces $\sim x = 1$ y D estará prendido.

Cuando el interruptor x está *abierto* $(x = 0)$, el circuito eléctrico incluye al foco D que, por lo tanto, está prendido. Al cerrar el interruptor, las cargas eléctricas usan el circuito más *corto* que contiene al interruptor cerrado y excluyen la que contiene al foco, D está apagado.

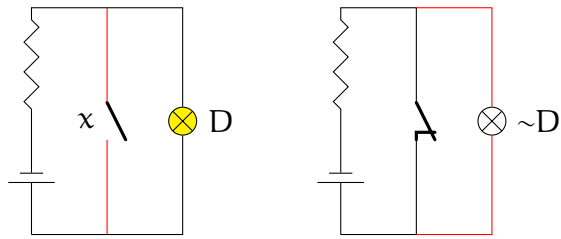


Figura 2.5 Circuito eléctrico del invertidor.

El estado del foco es el negativo del estado del interruptor.

A dicho dispositivo se le llama un ***invertidor***.

El nuevo elemento que aparece en el circuito, ⟋⟍⟋⟍, es una ***resistencia***, que impide la conexión directa entre los polos del potencial, evitando un ***corto circuito***. Añadiremos una *resistencia* donde exista la posibilidad de un corto ciruito.

Actividad 2.1. Con una pila, cables, interruptores y foco realiza el esquema del ***apagador*** de la Figura 2.1 y del ***invertidor*** de la Figura 2.5. ¿Son equivalentes? Compara el consumo de energía eléctrica en cada caso. De ser necesario, usa alguna resistencia.

2.2. Intersección, conjunción, AND

La ***intersección*** $A \cap B$ de dos conjuntos, es el conjunto de objetos que pertenecen a A **y** a B,

$$A \cap B = \{x \in \Omega \mid x \in A \ \text{ y } \ x \in B\},$$

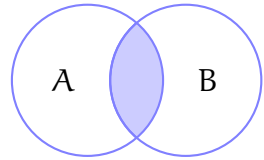


Figura 2.6 La parte sombreada representa la ***intersección***.

Ejemplo 2.1 Sean los conjuntos $A = \{2,3,5,6\}$, $B = \{1,2,4,5\}$, $C = \{1,3,5\}$.

Tenemos que $A \cap B = \{2,5\}$, $A \cap C = \{3,5\}$. ☺

Ejemplo 2.2 Si X es el conjunto de múltiplos positivos de 4 menores que 10 y Y es el conjunto de múltiplos positivos de 2 menores que 10, demuestra que $X \cap Y = X$.

Solución. Una manera de verificar la afirmación es listar los dos conjuntos y efectuar la operación de *intersección*,

$$X = \{4,8\}$$
$$Y = \{2,4,6,8\}$$

luego

$$X \cap Y = \{4,8\} = X.$$ ☺

Ajenos

Dos conjuntos A, B, son ***ajenos*** si su intersección es el conjunto vacío, es decir

A y B son *ajenos* si, y sólo si, $A \cap B = \emptyset$

Ejemplo 2.3 El mejor ejemplo de conjuntos ajenos es A y su complemento, $A \cap A^c = \emptyset$. Dado $x \in A$ tenemos que $x \notin A^c$, *viceversa*, si $x \in A^c$ por definición $x \notin A$, así A y A^c no tienen elementos en común. ☺

Ejemplo 2.4 Sea Ω el conjunto de los estados de un dispositivo de cuatro bits. Considera el conjunto A de los estados de Ω que tienen prendido el primero y tercer bit, B el conjunto de los estados que tienen prendido el tercer y cuarto bit. ¿Cuál es la intersección de A y B?

Solución. El conjunto Ω es

$$\begin{aligned}\Omega = \{&\texttt{0000}, \texttt{0001}, \texttt{0010}, \texttt{0011},\\ &\texttt{0100}, \texttt{0101}, \texttt{0110}, \texttt{0111},\\ &\texttt{1000}, \texttt{1001}, \texttt{1010}, \texttt{1011},\\ &\texttt{1100}, \texttt{1101}, \texttt{1110}, \texttt{1111}\}\end{aligned}$$

El conjunto A con el primer y tercer bit prendido es

$$A = \{\texttt{0101}, \texttt{0111}, \texttt{1101}, \texttt{1111}\}$$

y el conjunto B, tercer y cuarto bit prendidos,

$$B = \{\texttt{1100}, \texttt{1101}, \texttt{1110}, \texttt{1111}\},$$

luego

$$A \cap B = \{\texttt{1101}, \texttt{1111}\}.$$

Conjunción

La ***conjunción*** $p \wedge q$ de dos proposiciones (se lee "p ***y*** q"), es otra proposición.

$p \wedge q$ es verdadera **si** p es verdadera **y** q es verdadera

Su *valor de verdad* está dado por la tabla siguiente, llamada la *tabla de verdad* de la conjunción:

p	q	p ∧ q
V	V	V
V	F	F
F	V	F
F	F	F

En la primera y segunda columna aparecen las posibles combinaciones de los valores de verdad de p, de q, en la tercera columna el valor correspondiente a la proposición $p \wedge q$ según la definición de *conjunción*. Por ejemplo, en el tercer renglón vemos que p es Falsa, q es Verdadera, luego, según la definición de *conjunción*, $p \wedge q$ es Falsa.

Ejemplo 2.5 Consideremos las siguientes afirmaciones:

p: 7 es par,
q: Santiago es la capital de Chile.

La conjunción de las proposiciones p, q, a saber,

7 es par **y** Santiago es la capital de Chile,

es falsa pues p es falsa (el 7 no es un número par). No importa que q sea verdadera (sabemos que es verdad que Santiago es la capital de Chile). Para que la conjunción de dos proposiciones sea verdadera **es necesario** que las dos proposiciones sean verdaderas. ☺

AND

La operación entre bits similar a la intersección de conjuntos y a la conjunción de proposiciones, es la operación AND, que se denota con &. Si x, y son los estados de dos bits, el estado del

bit x & y es 1 sólo cuando el estado de **cada uno** de los bits es 1. Se ilustra en la tabla siguiente,

x	y	x & y
0	0	0
0	1	0
1	0	0
1	1	1

En el caso de los circuitos eléctricos, ilustramos la operación AND con dos interruptores colocados de manera tal que pasa la corriente sólo si **ambos** están cerrados,

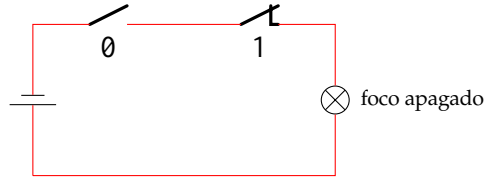


Figura 2.7 Con un interruptor abierto no pasa la corriente.

Cuando en un circuito están colocados dos interruptores, uno a continuación del otro, se dice que están ***en serie***

En el caso de un dispositivo de varios bits, digamos de un *byte*, la operación AND se realiza **bit a bit**, es decir,

$$0111\,0010 \;\&\; 0110\,1011 = 0110\,0010$$

O, puesto de otra manera,

$$\begin{aligned} & 0111\,0010 \\ \text{AND}\ & 0110\,1011 \\ = & \; 0110\,0010 \end{aligned}$$

Ejemplo 2.6 En un dispositivo de cuatro bits, tenemos una pareja de estados. Di cuáles son los bits prendidos ambos estados.

$$0111, \quad 1110$$

Solución. Aplicamos la operación AND para ver cuáles son los bits prendidos en *ambos* estados.

$$\begin{aligned} & 0111 \\ \text{AND } & 1110 \\ = \; & 0110 \end{aligned}$$

Luego los bits prendidos en ambos estados son el segundo y tercer bit, es decir, el bit 1 y el 2 (recuerda que los bits se cuentan de derecha a izquierda). ☺

Sea un byte xxxx xxxx, queremos saber si están prendidos los bits 2 y 5; realizamos la operación AND con un byte que tenga sólo esos bits prendidos. Si en el resultado están prendidos los bits 2 y 5, también lo están en el byte original, es decir, si

$$\begin{aligned} & \text{xxxx xxxx} \\ \text{AND } & 0010\,0100 \\ = \; & \text{yy1y y1yy}, \end{aligned}$$

entonces sabremos que el byte original es de la forma xx1x x1xx, es decir, que los bits 2 y 5 están encendidos. El byte 0010 0100 actúa como una *máscara* que sólo deja ver los bits 2 y 5 del byte xxxx xxxx.

Compuerta lógica AND

El dispositivo lógico que realiza la operación & es la ***compuerta lógica*** AND y se representa con el símbolo  .

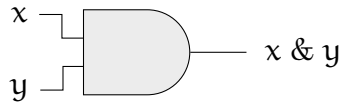


Figura 2.8 *Compuerta lógica* AND.

La Figura 2.7 de la página 22 representa el circuito eléctrico de la compuerta AND. La salida $x \,\&\, y$ es 1 sí, y sólo si, *ambas* entradas x, y son 1, es decir (el símbolo $\Leftrightarrow$ se lee *si, y sólo si,*),

$$x \,\&\, y = 1 \quad \Leftrightarrow \quad x = 1 \ \wedge \ y = 1.$$

Actividad 2.2. Construye un dispositivo con una pila, cables, dos interruptores, un foco y, de ser necesario, resistencias, de manera tal que para prender el foco, deban estar cerrados los dos interruptores. Es decir, si se abre *algún* interruptor, se apaga.

Problema 2.1 ¿Cuál es la tabla de verdad de $\sim(x \,\&\, y)$?

2.3. Unión, disyunción, OR

La ***unión*** $A \cup B$ de dos conjuntos, es el conjunto de elementos que pertenecen a A o a B (o a *ambos*),

$$A \cup B = \{\, x \in \Omega \mid x \in A \ \text{ o } \ x \in B \,\},$$

Ejemplo 2.7 Sea $\Omega = \{1, 2, 3, 4, 5, 6\}$, los conjuntos $A = \{2, 3, 5, 6\}$, $B = \{1, 2, 4, 5\}$, $C = \{1, 3, 5\}$.

Tenemos que $A \cup C = \{1, 2, 3, 5, 6\}$, $A \cup B = \Omega$. ☺

Ejemplo 2.8 Si X es el conjunto de múltiplos positivos de 4 menores que 10 y Y es el conjunto de múltiplos positivos de 2 menores que 10, demuestra que $X \cup Y = Y$.

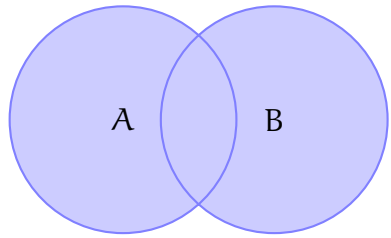


Figura 2.9 La parte sombreada representa la ***unión***.

Solución. Una manera de verificar la afirmación es listar los dos conjuntos y efectuar la operación de *unión*,

$$X = \{4, 8\}$$
$$Y = \{2, 4, 6, 8\}$$

luego

$$X \cup Y = \{2, 4, 6, 8\} = Y.$$ ☺

Para pertenecer a la *unión* de dos conjuntos, un objeto debe pertenecer a *al menos uno* de los dos (puede pertenecer a *ambos*).

Disyunción

La ***disyunción*** $p \vee q$ de dos proposiciones (se lee "p ***o*** q"), es otra proposición.

$p \vee q$ es verdadera **si** p es verdadera **o** q es verdadera

Su *valor de verdad* está dado por la tabla siguiente, llamada la *tabla de verdad* de la disyunción:

p	q	p ∨ q
V	V	V
V	F	V
F	V	V
F	F	F

En la primera y segunda columna aparecen las posibles combinaciones de los valores de verdad de p, de q, en la tercera columna el valor correspondiente a la proposición $p \vee q$ según la definición de *disyunción*. Por ejemplo, en el tercer renglón vemos que p es Falsa, q es Verdadera, luego, según la definición de *disyunción*, $p \vee q$ es Verdadera.

Ejemplo 2.9 Sean las proposiciones p y q las siguientes:

p: 6 es par,
q: 6 es múltiplo de 3.

La disyunción $p \vee q$ es verdadera —para que sea verdadera **basta** que una de las proposiciones lo sea— pues sucede que, en este caso, las dos proposiciones son verdaderas. ☺

OR

La operación entre bits similar a la intersección de conjuntos y a la disyunción de proposiciones, es la operación OR, que se denota con |. Si x, y, son los estados de dos bits, el estado del bit $x \mid y$ es 1 cuando el estado de **alguno** de los dos bits es 1. Se ilustra en la tabla siguiente

x	y	x | y
0	0	0
0	1	1
1	0	1
1	1	1

En el caso de los circuitos eléctricos, ilustramos la operación OR con dos interruptores colocados de manera tal que pasa la corriente si **alguno** está cerrado,

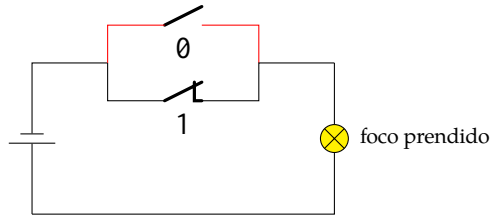


Figura 2.10 Basta un interruptor *cerrado* para que pase la corriente.

Se dice que los interruptores colocados como en la figura anterior, están en ***paralelo***.

En el caso de un dispositivo de varios bits, digamos de un *byte*, la operación OR también se realiza **bit a bit**, es decir,

$$0111\,0010 \mid 0110\,1011 = 0111\,1011$$

O, puesto de otra manera,

$$\begin{aligned} &0111\,0010\\ \text{OR}\ &0110\,1011\\ =\ &0111\,1011 \end{aligned}$$

Ejemplo 2.10 En un dispositivo de cuatro bits, tenemos una pareja de estados. Di cuáles son los bits prendidos en alguno de los estados.

$$0101, \quad 1100$$

Solución. Aplicamos la operación OR para ver cuáles son los bits prendidos en *algún* estado.

$$\begin{aligned} &0101\\ \text{OR}\ &1100\\ =\ &1101 \end{aligned}$$

Luego los bits que están prendidos en *alguno* de los estados son el primero, tercero y cuarto bit, es decir, los bits 0, 2 y 3 (recuerda que los bits se cuentan de derecha a izquierda). ☺

Sea un byte `xxxx xxxx`, queremos *prender* los bits `2` y `5`; realizamos la operación `OR` con un byte que tenga sólo esos bits prendidos. En el resultado **necesariamente** están prendidos los bits `2` y `5`,

$$\begin{aligned} & \texttt{xxxx xxxx} \\ \texttt{AND}\ & \texttt{0010 0100} \\ = & \texttt{xx1x x1xx}. \end{aligned}$$

Así, sin importar la forma del byte original, en el resultado los bits `2` y `5` están encendidos. Nuevamente, el byte `0010 0100` actúa como una *máscara* que sólo deja ver los bits `2` y `5` del byte `xxxx xxxx`.

Compuerta lógica `OR`

El dispositivo lógico que realiza la operación $|$ es la ***compuerta lógica*** `OR` y se representa con el símbolo  .

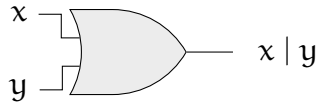


Figura 2.11 *Compuerta lógica* `OR`.

La Figura 2.10 de la página 27 representa el circuito eléctrico de la compuerta `OR`. La salida $x \mid y$ es `1` si *alguna* entrada x ó y es `1`, es decir (recuerda, el símbolo $\Leftrightarrow$ se lee *si, y sólo si,*),

$$x \mid y = 1 \quad \Leftrightarrow \quad x = 1 \ \vee \ y = 1.$$

Actividad 2.3. Construye un dispositivo con una pila, cables, dos interruptores, un foco y, de ser necesario, resistencias, de manera tal que para prender el foco, baste cerrar un interruptor. Es decir, si se cierra *algún* interruptor, se prende el foco.

Problema 2.2 ¿Cuál es la tabla de verdad de $\sim(x \mid y)$?

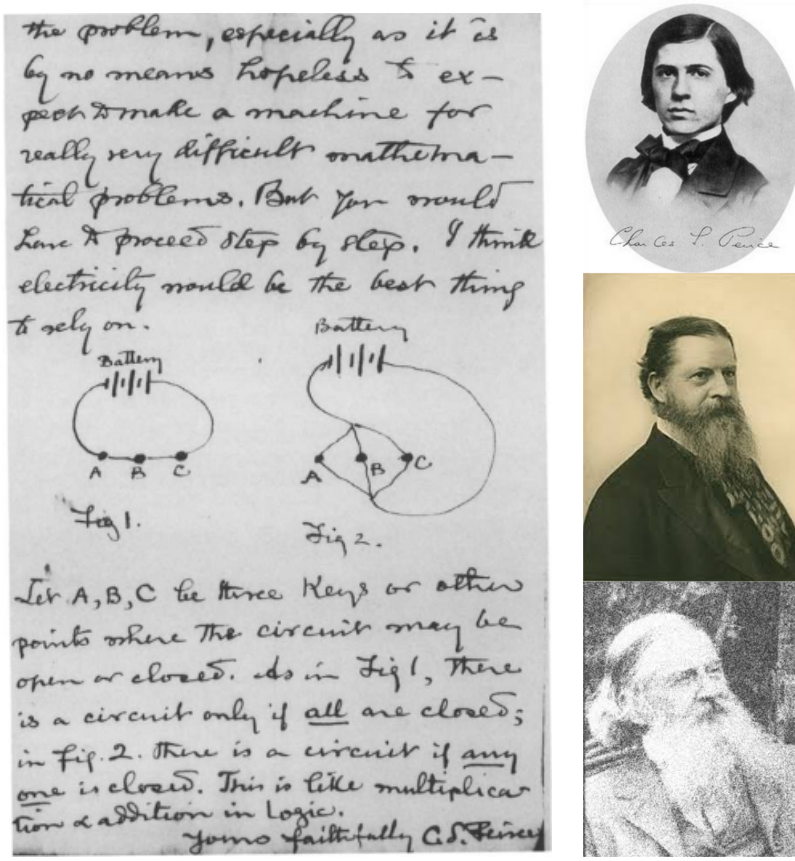


Figura 2.12 Carta a Marquand en *Writings of Charles S. Peirce*, páginas 421–23. A la derecha, Peirce en varias etapas de su vida.

Texto de la carta mostrada en la Figura 2.12, página 29

Creo que deberías retomar / el problema, sobre todo porque de ninguna manera es imposible construir una máquina para resolver problemas matemáticos realmente muy difíciles. Pero tendrías que proceder paso a paso. Creo que la electricidad es la mejor base.

> *Sean A, B, C tres interruptores o puntos donde el circuito pueda estar abierto o cerrado. Como en la Fig. 1, hay un circuito sólo si todos están cerrados; en la Fig. 2. hay un circuito si alguno está cerrado. Esto es como la multiplicación y la suma en Lógica.*
>
> *Recibe un cordial saludo*
>
> *Charles Sanders Peirce*

2.4. Diferencia

Consideremos a dos conjuntos A, B. La operación que representa a los elementos de A que ***no*** están en B se llama la ***diferencia*** $A \setminus B$, se lee A *diferencia* B,

$$A \setminus B = \{ x \in A \mid x \notin B \}.$$

Ejemplo 2.11 Sea A el conjunto de las personas a quienes gusta la interpretación de Yuja Wang de los *24 Preludi, Op. 28* de Frédéric Chopin, B el conjunto de las personas a quienes gusta la pieza *Wolf Totem* de la banda mongola de *heavy metal*, *The HU*. Describe los conjuntos $A \setminus B$ y $B \setminus A$.

Solución. $A \setminus B$ es el conjunto de personas a quienes gusta la interpretación de Yuja Wang de los Preludios de Chopin **pero no** les gusta la pieza *Wolf Totem*, de *The HU*.

$B \setminus A$ es el conjunto de personas a quienes gusta la pieza *Wolf Totem*, de *The HU*, **pero no** les gusta la interpretación de Yuja Wang de los Preludios de Chopin. ☺

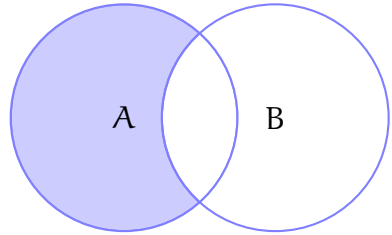


Figura 2.13 $A \setminus B$, los elementos de A que **no están** en B.

En *proposiciones*, la operación equivalente a la *diferencia* entre conjuntos, es la definida por $p \wedge \neg q$, cuya tabla de verdad es

p	q	$\neg q$	$p \wedge \neg q$
V	V	F	F
V	F	V	V
F	V	F	F
F	F	V	F

Para *bits*, la operación equivalente está definida por $x \,\&\, \sim y$, con tabla de verdad

x	y	$\sim y$	$x \,\&\, \sim y$
0	0	1	0
0	1	0	0
1	0	1	1
1	1	0	0

Lo cual, en circuitos eléctricos, nos dice que el *foco* está prendido sólo cuando el interruptor x está cerrado **y** el y está abierto.

Añadimos una *resistencia* para evitar un *corto circuito* al cerrar el interruptor y.

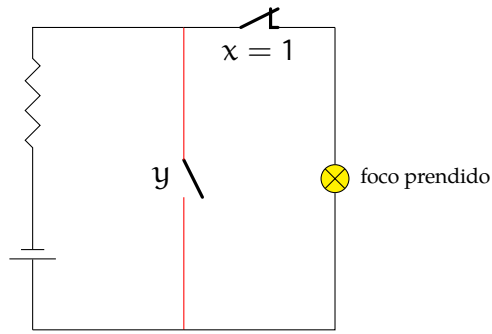


Figura 2.14 Al cerrar y se apaga el foco.

La siguiente figura corresponde al circuito lógico

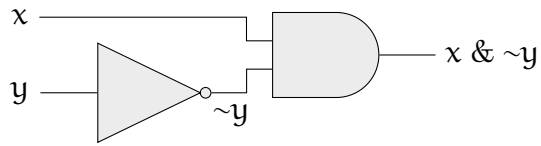


Figura 2.15 Circuito lógico de la *diferencia*.

El valor de x & ~y está determinado por la tabla de verdad al final de la página anterior; la salida es 1 sólo cuando $x = 1$ y $y = 0$.

Actividad 2.4. Con pilas, cables, interruptores, foco y de ser necesario, resistencias, así como en las **Actividades** anteriores, construye un dispositivo que tenga el foco prendido según la tabla de verdad de x & ~y.

2.5. Diferencia simétrica, disyunción excluyente, XOR

Sean A, B, dos conjuntos. La ***diferencia simétrica*** $A \triangle B$, es la *unión* de $A \setminus B$ con $B \setminus A$, es el conjunto de puntos que están en A o en B pero **no** en ambos, es decir

$$A \triangle B = (A \setminus B) \cup (B \setminus A).$$

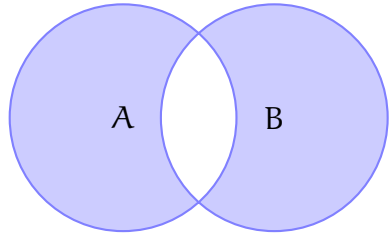


Figura 2.16 *A diferencia simétrica* B.

La definición anterior de diferencia simétrica equivale a

$$A \triangle B = (A \cup B) \setminus (A \cap B).$$

La diferencia simétrica es
la unión *menos* la intersección

Ejemplo 2.12 Usemos los conjuntos A, B, del Ejemplo 2.11, página 30, sobre Chopin y *The HU*. La diferencia simétrica $A \triangle B$ es el conjunto de las personas que gustan de la interpretación de Yuja o de *Wolf Totem*, pero **no de ambas**. Es decir, consideramos la unión de A y B, pero excluimos la intersección $A \cap B$. ☺

Disyunción excluyente

La ***disyunción excluyente*** se denota con $p \veebar q$ (se lee "***una de dos,*** p ***ó*** q"), es otra proposición, donde p, q, son proposiciones.

$p \veebar q$ es verdadera **si** p es verdadera **o** q es verdadera, ***y* es falso** que p es verdadera ***y*** q es verdadera

Es decir, $p \veebar q$ tiene el mismo valor de verdad que

$$(p \vee q) \wedge \big(\neg(p \wedge q)\big).$$

Su *valor de verdad* está dado por la tabla siguiente[1], llamada la *tabla de verdad* de la disyunción excluyente:

p	q	p $\veebar$ q
V	V	F
V	F	V
F	V	V
F	F	F

Vemos que el valor de verdad de p $\veebar$ q es V, en el segundo y tercer renglón, sólo cuando p verdadera y q falsa ***o*** cuando p es falsa y q verdadera. El valor de verdad de p $\veebar$ q es F, en el primer y cuarto renglón, cuando ambas, p, q, son verdaderas, o ambas son falsas.

Ejemplo 2.13 Si p y q son las siguientes proposiciones,

p: 20 es múltiplo de 5,
q: 20 es par,

Tenemos que p es verdadera pues, en efecto, el número 20 es múltiplo de 5 porque $20 = 5 \times 4$. Asimismo q es verdadera pues $20 = 2 \times 10$. Entonces la proposición p $\veebar$ q

p $\veebar$ q: 20 es, una de dos, múltiplo de 5 o es par,

es falsa. ☺

En el lenguaje cotidiano usamos de manera indistinta la disyunción y la disyunción excluyente, del contexto distinguimos el sentido utilizado.

Actividad 2.5. Con un grupo de personas da ejemplos de disyunciones, como ¿subes o bajas?, ¿postre o café?, ¿café o té?, y analiza el sentido con el cual se usa, excluyente o no.

[1] Ver la construcción del tabla de verdad de la disyunción excluyente en *Conjuntos, lógica y funciones*, pp. 54–55

XOR

La operación entre bits, similar a la diferencia simétrica de conjuntos y a la disyunción excluyente de proposiciones, es la operación XOR, que se denota con ^. Si x, y, son los estados de dos bits, el estado del bit x ^ y es 1 cuando el estado de **sólo uno** de los dos bits es 1, como se muestra en la tabla siguiente

x	y	x ^ y
0	0	0
0	1	1
1	0	1
1	1	0

Ilustramos la operación XOR con circuitos eléctricos, con cuatro interruptores, acoplados dos a dos de modo que el acoplado tendrá el negativo del valor.

Supongamos que x, y, son los interruptores, los acoplados tendrán los valores ~x, ~y, respectivamente. Es decir, si un interruptor está abierto, su acoplado estará cerrado, y viceversa.

Por un lado colocamos en *serie* x, ~y, es decir x & ~y; por otro lado colocamos ~x, y, es decir, ~x & y.

Ahora colocamos estos dos en *paralelo*, es decir

$$(x \,\&\, \sim y) \mid (\sim x \,\&\, y) = x \,\hat{}\, y.$$

Están colocados de manera tal que pasa la corriente si **sólo uno** está cerrado, según se ve en la *tabla de verdad*.

En las figuras siguientes veremos el comportamiento del circuito eléctrico asociado.

Recuerda que son dos interruptores independientes, x, y; hay otros dos interruptores que están acoplados a los primeros y se comportan, uno como ~x, el otro como ~y.

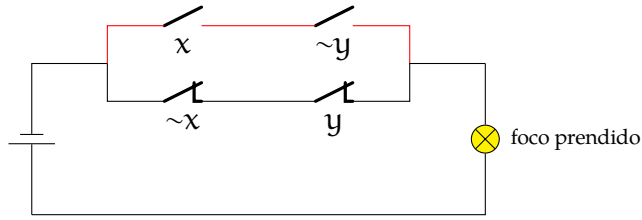


Figura 2.17 y está *cerrado*, x está abierto; pasa la corriente.

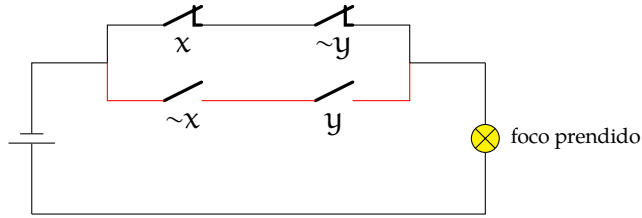


Figura 2.18 x está *cerrado*, y está abierto; pasa la corriente.

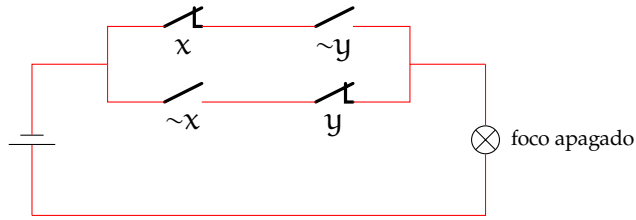


Figura 2.19 Ambos están *cerrados*; no pasa la corriente.

Como en las operaciones anteriores, en el caso de un dispositivo de varios bits, digamos de un *byte*, la operación `XOR` se realiza **bit a bit**, es decir,

$$\texttt{0111 0010 ^ 0110 1011} = \texttt{0001 1001}$$

O, puesto de otra manera,

$$\begin{aligned} & \texttt{0111 0010} \\ \texttt{XOR}\ & \texttt{0110 1011} \\ = & \texttt{0001 1001} \end{aligned}$$

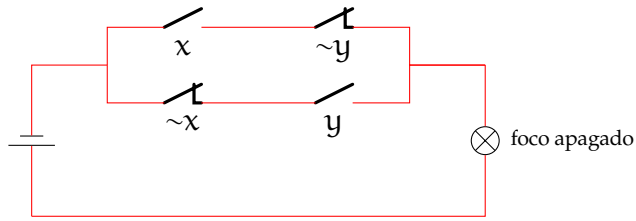


Figura 2.20 Ambos están *abiertos*; no pasa la corriente.

Ejemplo 2.14 En un dispositivo de cuatro bits, tenemos una pareja de estados. Di cuáles son los bits que están prendidos en algún estado pero no en ambos.

$$0111, \quad 1110$$

Solución. Aplicamos la operación XOR para ver cuáles son los bits prendidos en algún estado pero no en *ambos*.

$$\begin{aligned} & 0111 \\ \texttt{XOR}\ & 1110 \\ = & 1001 \end{aligned}$$

Luego los bits prendidos en algún estado son todos, pero sólo el primero y cuarto bit, es decir, el bit 0 y el 3 (recuerda que los bits se cuentan de derecha a izquierda) no están prendidos en *ambos*, el bit 0 sólo está prendido en el primer estado y el bit 3 está prendido sólo en el segundo. ☺

Sea 1110, queremos cambiar el valor de los bits 0 y 3; realizamos la operación XOR con el estado que tenga sólo esos bits prendidos.

$$\begin{aligned} & 1110 \\ \texttt{XOR}\ & 1001 \\ = & 0111, \end{aligned}$$

en el resultado vemos el cambio en el valor de los bits 0 y 3.

Compuerta lógica XOR

El dispositivo lógico que realiza la operación ^ es la ***compuerta lógica*** XOR y se representa con el símbolo  .

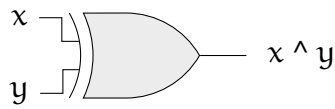


Figura 2.21 *Compuerta lógica* XOR.

La Figura 2.17 de la página 36 representa un comportamiento del circuito eléctrico de la compuerta XOR. La salida x ^ y es 1 si *alguna* entrada x ó y es 1, pero no *ambas*, es decir (recuerda, el símbolo $\Leftrightarrow$ se lee *si, y sólo si,*),

$$x \wedge y = 1 \quad \Leftrightarrow \quad x = 1 \ \veebar \ y = 1.$$

Ilustramos el comportamiento lógico de la compuerta XOR en la figura siguiente.

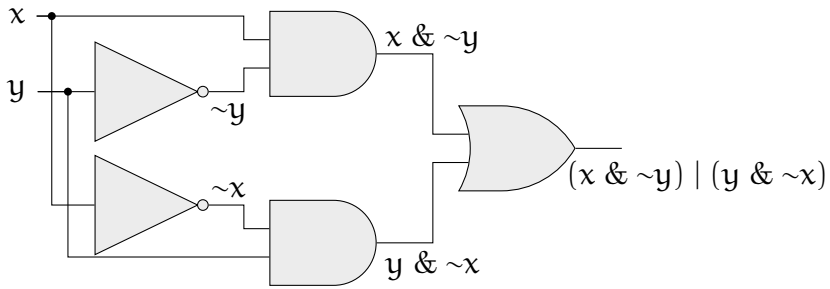


Figura 2.22 Circuito lógico de XOR.

Actividad 2.6. Como en las compuertas anteriores, con pilas, interruptores, focos y de ser necesario, resistencias, construye un dispositivo que prenda el foco cuando solamente alguno de los interruptores esté cerrado.

2.6. Leyes de De Morgan

En el Capítulo 3 veremos las propiedades de las operaciones similares entre conjuntos, proposiciones, bits y compuertas lógicas. En esta Sección, tratamos con unas propiedades comunes a esos conceptos que permiten construir nuevas compuertas lógicas básicas. Las enunciamos ahora en su versión para el lenguaje de los conjuntos. Ver la demostración en *Conjuntos, lógica y funciones*, pp. 34–35.

Leyes de De Morgan. Sean A, B, dos conjuntos, se tiene que

1. $(A \cap B)^c = A^c \cup B^c$,
2. $(A \cup B)^c = A^c \cap B^c$.

La primera se lee:

> el complemento de la **intersección** es
> la **unión** de los complementos

y la segunda,

> el complemento de la **unión** es
> la **intersección** de los complementos

Complemento de la intersección, NAND

En el lenguaje de conjuntos la primera ley se expresa como

$$(A \cap B)^c = A^c \cup B^c.$$

La parte sombreada de la Figura 2.23 ilustra el complemento de la intersección. Primero se aplica la intersección y a ese resultado se aplica el complemento.

El análogo en proposiciones es aplicar primero la *conjunción* y al resultado se aplica la *negación*.

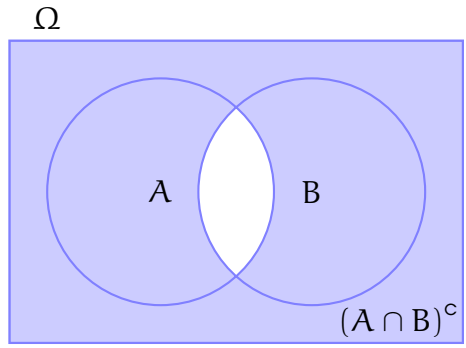


Figura 2.23 Primero la intersección y luego el complemento

La versión para **proposiciones** de la primera ley es:

Ley de De Morgan 1. Sean dos *proposiciones*, p, q, se tiene que

$$\neg(p \wedge q) \equiv \neg p \vee \neg q,$$

que se lee, *la negación de una conjunción es* ***equivalente*** *a la disyunción de las negaciones*.

En la expresión anterior, la palabra ***equivalente*** significa que ambas expresiones, a la izquierda y derecha del símbolo $\equiv$, tienen la misma tabla de verdad.

Ejemplo 2.15 Verifica que las tablas de verdad de la equivalencia anterior, son iguales.

Solución. Construimos la tabla de verdad para ambos lados de la equivalencia, $\neg(p \wedge q) \equiv \neg p \vee \neg q$, primero el lado izquierdo,

p	q	$p \wedge q$	$\neg(p \wedge q)$
V	V	V	F
V	F	F	V
F	V	F	V
F	F	F	V

después el lado derecho,

p	q	$\neg p$	$\neg q$	$\neg p \vee \neg q$
V	V	F	F	F
V	F	F	V	V
F	V	V	F	V
F	F	V	V	V

Los valores de verdad de la última columna de cada tabla son iguales. ☺

NAND

La operación entre bits, similar al complemento de la intersección de dos conjuntos, y a la negación de la conjunción de dos proposiciones, es la operación NAND. Si x, y, son los estados de dos bits, el estado del bit x NAND y es 1 en todos los casos, **excepto** cuando el estado de **los dos** bits es 1. Se ilustra en la tabla siguiente

x	y	x NAND y
0	0	1
0	1	1
1	0	1
1	1	0

La versión para **bits** de la primera ley de De Morgan es:

Ley de De Morgan 1. Sean x y y los estados de dos *bits*, se tiene

$$\sim(x \,\&\, y) = \sim x \mid \sim y,$$

que se lee *la negación de la operación* AND *es* ***igual*** *a la operación* OR *de las negaciones*.

En la expresión anterior, la palabra ***igual*** significa que ambas expresiones, a la izquierda y derecha del símbolo $\equiv$, tienen la misma tabla de verdad.

En el caso de los circuitos eléctricos, ilustramos la operación NAND con dos interruptores colocados de manera tal que siempre pasa la corriente excepto si **ambos** están cerrados,

Nota que la tabla NAND es la negación de la tabla AND,

$$1110 = \sim 0001.$$

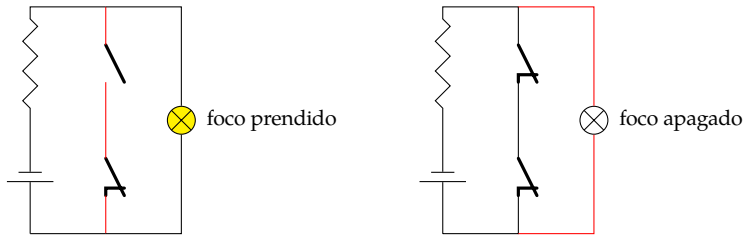


Figura 2.24 NAND, *prendido* excepto cuando los **dos** interruptores están *cerrados*.

Problema 2.3 Dibuja el diagrama del circuito eléctrico NAND en el caso en que $x = y = 0$. ¿Está prendido el foco?

En el caso de un dispositivo de varios bits, digamos de un *byte*, la operación NAND se realiza **bit a bit**, es decir,

$$0111\ 0010 \text{ NAND } 0110\ 1011 = 1001\ 1101$$

O, puesto de otra manera,

$$\begin{array}{r} 0111\ 0010 \\ \text{NAND } 0110\ 1011 \\ = 1001\ 1101 \end{array}$$

Ejemplo 2.16 En un dispositivo de cuatro bits, tenemos una pareja de estados. Di cuáles son los bits prendidos ambos estados.

$$0111, \quad 1110$$

Solución. En el caso del Ejemplo 2.6 en la página 23 aplicamos la operación AND, los bits prendidos del resultado indicaron los bits prendidos en *ambos* estados. En tanto que NAND es la negación de AND, ahora los bits *apagados* en el resultado, indican los bits prendidos en *ambos* estados.

$$\begin{aligned} & 0111 \\ \text{NAND } & 1110 \\ = & \ 1001 \end{aligned}$$

En el resultado, los bits 1 y 2 están *apagados*, luego los bits *prendidos* en ambos estados son el segundo y tercer bit. ☺

Sea un byte xxxx xxxx, queremos saber si están prendidos los bits 2 y 5; realizamos la operación NAND con un byte que tenga sólo esos bits prendidos. Si en el resultado están *apagados* los bits 2 y 5, significa que están *prendidos* en el byte original, es decir, si

$$\begin{aligned} & xxxx\ xxxx \\ \text{NAND } & 0010\ 0100 \\ = & \ yy0y\ y0yy, \end{aligned}$$

entonces sabremos que el byte original es de la forma xx1x x1xx, es decir, que los bits 2 y 5 están encendidos. El byte 0010 0100 actúa como una *máscara* (en este caso como una máscara en *negativo*) que sólo deja ver los bits 2 y 5 del byte xxxx xxxx.

Compuerta lógica NAND

La Figura 2.24 de la página 42 representa el circuito eléctrico de la compuerta NAND. La salida x NAND y es 0 sí, y sólo si, *ambas* entradas, x, y, son 1, es decir (el símbolo $\Leftrightarrow$ se lee *si, y sólo si,*),

$$x \text{ NAND } y = 0 \Leftrightarrow x = 1 \text{ y } y = 1.$$

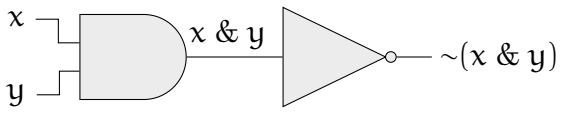


Figura 2.25 Circuito lógico de NAND.

Por la primera ley de De Morgan, el circuito lógico anterior es equivalente al siguiente:

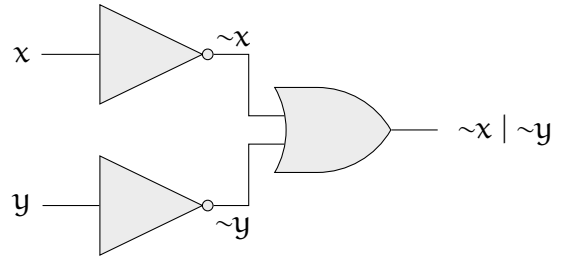


Figura 2.26 $\sim(x \,\&\, y) = \sim x \mid \sim y$.

El dispositivo lógico que realiza la operación NAND es la ***compuerta lógica***  .

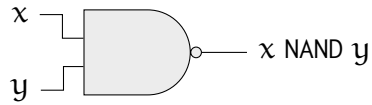


Figura 2.27 *Compuerta lógica* NAND.

Nota que el símbolo para NAND difiere del asignado a AND en una pequeña *burbuja* al final.

Actividad 2.7. Construye un dispositivo con una pila, cables, dos interruptores, un foco y, de ser necesario, resistencias, de manera tal que el foco se apague cuando se cierren los dos interruptores. Es decir, si se abre *algún* interruptor (o ambos), el foco se prende.

Complemento de la unión, NOR

En el lenguaje de conjuntos la segunda ley se expresa como

$$(A \cup B)^c = A^c \cap B^c.$$

La parte sombreada de la Figura 2.28 ilustra el complemento de la unión. Primero se aplica la unión y después, a ese resultado se aplica el complemento.

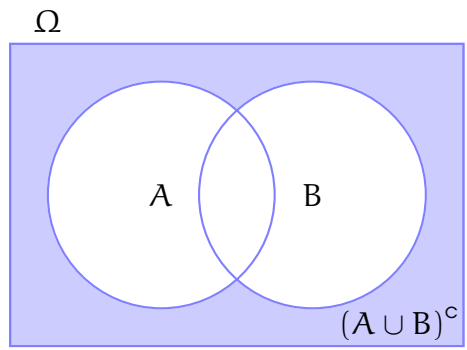


Figura 2.28 Primero la unión y luego el complemento

El análogo en proposiciones es aplicar primero la *disyunción* y al resultado se aplica la *negación*.

La versión para **proposiciones** de la segunda ley es:

Ley de De Morgan 2. Sean dos *proposiciones*, p, q, se tiene que

$$\neg(p \vee q) \equiv \neg p \wedge \neg q,$$

que se lee *la negación de una disyunción es **equivalente** a la conjunción de las negaciones*.

En la expresión anterior, la palabra ***equivalente*** significa que ambas expresiones, a la izquierda y derecha del símbolo $\equiv$, tienen la misma tabla de verdad.

Ejemplo 2.17 Verifica que las tablas de verdad de la equivalencia anterior, son iguales.

Solución. Construimos la tabla de verdad para ambos lados de la equivalencia, $\neg(p \vee q) \equiv \neg p \wedge \neg q$, primero el lado izquierdo,

p	q	$p \vee q$	$\neg(p \vee q)$
V	V	V	F
V	F	V	F
F	V	V	F
F	F	F	V

después el lado derecho,

p	q	$\neg p$	$\neg q$	$\neg p \wedge \neg q$
V	V	F	F	F
V	F	F	V	F
F	V	V	F	F
F	F	V	V	V

Los valores de verdad de la última columna de cada tabla son iguales. ☺

NOR

La operación similar al complemento de la unión de dos conjuntos, y a la negación de la disyunción de dos proposiciones, es la operación NOR, entre bits. Si x, y, son los estados de dos bits,

el estado del bit x NOR y es 1 **sólo** cuando el estado de **los dos** bits es 0. Se ilustra en la tabla siguiente

x	y	x NOR y
0	0	1
0	1	0
1	0	0
1	1	0

La versión para **bits** de la segunda ley de De Morgan es:

Ley de De Morgan 2. Sean x, y, los estados de dos *bits*, se tiene

$$\sim(x \mid y) = \sim x \,\&\, \sim y,$$

que se lee *la negación de la operación* OR *es* ***igual*** *a la operación* AND *de las negaciones*.

En la expresión anterior, la palabra ***igual*** significa que ambas expresiones, a la izquierda y derecha del símbolo $=$, tienen la misma tabla de verdad.

En el caso de los circuitos eléctricos, ilustramos la operación NOR con dos interruptores colocados de manera tal que pasa la corriente *sólo* si **ambos** están abiertos,

Nota que la la tabla de NOR es la negación de la tabla de OR,

$$1000 = \sim 0111.$$

Problema 2.4 Dibuja el diagrama del circuito eléctrico NOR en el caso en que $x = y = 1$. ¿Está prendido el foco?

En el caso de un dispositivo de varios bits, digamos de un *byte*, la operación NOR se realiza **bit a bit**, es decir,

$$0111\,0010 \text{ NOR } 0110\,1011 = 1000\,0100$$

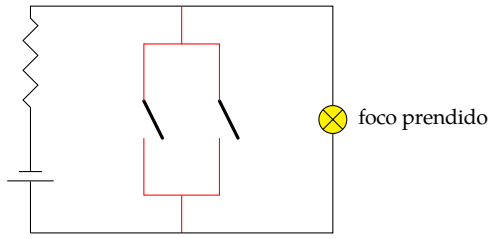


Figura 2.29 *Prende* sólo cuando los **dos** interruptores están *abiertos*.

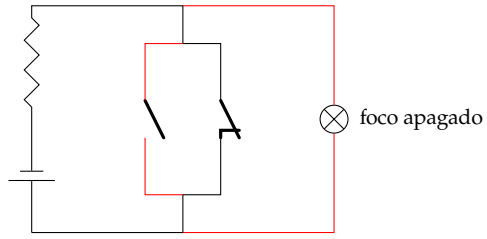


Figura 2.30 Si se *cierra* algún interruptor se *apaga* el foco.

O, puesto de otra manera,

$$\begin{aligned} & 0111\,0010 \\ \texttt{NOR}\ & 0110\,1011 \\ = {} & 1000\,0100 \end{aligned}$$

Ejemplo 2.18 En un dispositivo de cuatro bits, tenemos una pareja de estados. Di cuáles son los bits *apagados* ambos estados.

$$0111, \quad 1110$$

Solución. Aplicamos la operación `NOR`, los bits prendidos del resultado indican los bits apagados en *ambos* estados.

$$\begin{aligned} & 0111 \\ \texttt{NOR}\ & 1110 \\ = {} & 0000 \end{aligned}$$

En el resultado, todos los bits están *apagados*, luego ***no*** hay una pareja de bits *apagados* en ambos estados. ☺

Sea un byte `xxxx xxxx`, queremos saber si están apagados los bits `2, 5`; realizamos la operación `NOR` con un byte que tenga sólo esos bits apagados. Si en el resultado están *prendidos* los bits `2, 5`, significa que están *apagados* en el byte original, es decir, si

$$\begin{aligned} &\texttt{xxxx xxxx} \\ \texttt{NOR}\ &\texttt{1101 1011} \\ =\ &\texttt{yy1y y1yy}, \end{aligned}$$

entonces sabremos que el byte original es de la forma `xx0x x0xx`, es decir, que los bits `2` y `5` están apagados. El byte `0010 0100` actúa como una *máscara* (en este caso como una máscara en *negativo*) que sólo deja ver los bits `2` y `5` del byte `xxxx xxxx`.

Compuerta lógica `NOR`

La Figura 2.29 de la página 48 representa el circuito eléctrico de la compuerta `NOR`. La salida x `NOR` y es `1` si, y sólo si, *ambas* entradas, x, y, son `0`, es decir (el símbolo $\Leftrightarrow$ se lee *si, y sólo si,*),

$$x \texttt{ NOR } y = 1 \quad \Leftrightarrow \quad x = 0 \ \wedge\ y = 0.$$

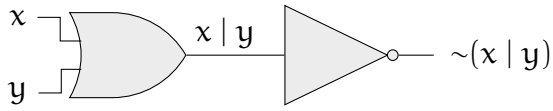


Figura 2.31 Circuito lógico de `NOR`.

Por la segunda ley de De Morgan, el circuito lógico anterior es equivalente al siguiente:

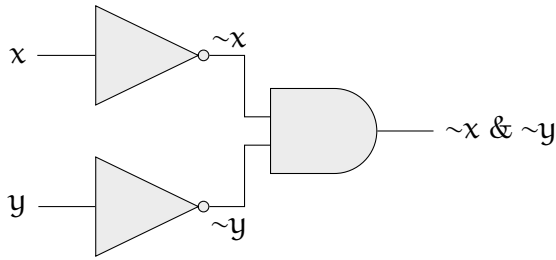


Figura 2.32 $\sim(x \mid y) = \sim x \,\&\, \sim y$.

El dispositivo lógico que realiza la operación NOR es la ***compuerta lógica***  .

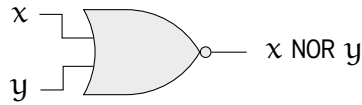


Figura 2.33 *Compuerta lógica* NOR.

Nota que el símbolo para NOR difiere del asignado a OR en una pequeña *burbuja* al final.

Actividad 2.8. Construye un dispositivo con una pila, cables, dos interruptores, un foco y, de ser necesario, resistencias, de manera tal que para que el foco prenda deban estar abiertos los dos interruptores. Es decir, si se cierra *algún* interruptor (o ambos), el foco se apaga.

2.7. Complemento de la diferencia simétrica, XNOR

Sean A, B, dos conjuntos. En la Sección 2.5, página 32, vimos que la ***diferencia simétrica*** de $A \triangle B$ es el conjunto de puntos que están en A o en B pero **no** en ambos, es decir

$$A \triangle B = (A \setminus B) \cup (B \setminus A),$$

que también podemos expresar como

$$A \triangle B = (A \cup B) \setminus (A \cap B).$$

Ahora vamos a construir una nueva operación lógica tomando el *complemento* de la *diferencia simétrica*

Analicemos el ***complemento de la diferencia simétrica*** de A y B aplicando las leyes de De Morgan (y algo de álgebra de conjuntos, ver *Conjuntos, lógica y funciones*).

$$\begin{aligned}
(A \triangle B)^c &= \big((A \setminus B) \cup (B \setminus A)\big)^c, \\
&= (A \setminus B)^c \cap (B \setminus A)^c, \\
&= (A \cap B^c)^c \cap (B \cap A^c)^c, \\
&= (A^c \cup B^{cc}) \cap (B^c \cup A^{cc}), \\
&= (A^c \cup B) \cap (B^c \cup A), \\
&= \big((A^c \cup B) \cap B^c\big) \cup \big((A^c \cup B) \cap A\big), \\
&= \big((A^c \cap B^c) \cup (B \cap B^c)\big) \cup \big((A^c \cap A) \cup (B \cap A)\big), \\
&= \big((A^c \cap B^c) \cup \emptyset\big) \cup \big(\emptyset \cup (B \cap A)\big), \\
&= (A^c \cap B^c) \cup (B \cap A).
\end{aligned}$$

Tenemos, finalmente, que

$$(A \triangle B)^c = (A^c \cap B^c) \cup (B \cap A).$$

Lo cual significa que el ***complemento de la diferencia simétrica*** es el conjunto de puntos que están en la intersección de los complementos de A y de B, ***o*** están en la intersección de A y B.

El *complemento de la diferencia simétrica* consta de los *puntos* que están en ***ambos*** conjuntos ***o*** en ***ninguno***.

La parte sombreada de la figura siguiente ilustra el conjunto.

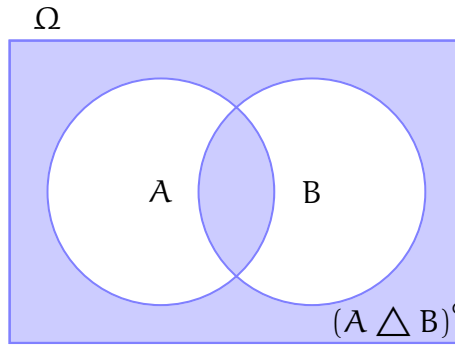


Figura 2.34 Complemento de la *diferencia simétrica*

Ejemplo 2.19 Usemos de nuevo los conjuntos A y B del Ejemplo 2.11, página 30, sobre Chopin y *The HU*. El *complemento de la diferencia simétrica* de A y B es el conjunto de las personas que gustan de **ambas** interpretaciones, la de Yuja y la de *Wolf Totem*, ***o*** de **ninguna**. Es decir, consideramos la unión de $A \cap B$ con $A^c \cap B^c$. ☺

Negación de la disyunción excluyente

La *proposición lógica* similar al *complemento de la diferencia simétrica* tiene un valor de verdad sugerido por su análogo en conjuntos,

$$\neg(p \veebar q) \equiv (\neg p \wedge \neg q) \vee (p \wedge q).$$

Aunque pudimos deducirlo por medio de operaciones lógicas similares a las realizadas con conjuntos.

De la equivalencia anterior desprendemos que la proposición en cuestión, la ***negación de la disyunción excluyente***, es verdadera sólo si **ambas**, p y q son verdaderas ***o*** **ambas** son falsas.

O simplemente, como lo indica la expresión $\neg(p \veebar q)$, el valor de verdad en la tabla es el negativo del valor de la tabla de $p \veebar q$.

Recordemos que $p \veebar q$ es verdadera sólo en los casos en que alguna es verdadera, pero no ambas.

Así, la tabla de verdad de $\neg(p \veebar q)$ es

p	q	$\neg(p \veebar q)$
V	V	V
V	F	F
F	V	F
F	F	V

XNOR

La operación entre bits similar al complemento de la diferencia simétrica de conjuntos y a la negación de la disyunción excluyente de proposiciones, es la operación XNOR. Si x, y, son los estados de dos bits, el estado del bit x XNOR y es `1` cuando el estado de **ambos** bits es `0` o el estado de **ambos** es `1`, como se muestra en la tabla siguiente

x	y	x **XNOR** y
`0`	`0`	`1`
`0`	`1`	`0`
`1`	`0`	`0`
`1`	`1`	`1`

Ilustramos la operación XNOR con circuitos eléctricos, con cuatro interruptores, acoplados dos a dos de modo que el acoplado tendrá el negativo del valor.

Supongamos que x, y, son los interruptores, los acoplados tendrán los valores $\sim x$, $\sim y$, respectivamente. Es decir, si un interruptor está abierto, su acoplado estará cerrado, y viceversa.

Por un lado colocamos x, y, en *serie*, es decir $x \,\&\, y$; por otro lado colocamos $\sim x$, $\sim y$, también en *serie* es decir, $\sim x \,\&\, \sim y$.

Ahora colocamos estos dos en *paralelo*, es decir

$$(x \,\&\, y) \mid (\sim x \,\&\, \sim y) = x \texttt{ XNOR } y.$$

Están colocados de manera tal que pasa la corriente si **ambos** están abiertos o **ambos** están cerrados, según se ve en la *tabla de verdad*.

En las figuras siguientes veremos el comportamiento del circuito eléctrico asociado.

Recuerda que son dos interruptores independientes, x, y; hay otros dos interruptores que están acoplados a los primeros: se comportan, uno como $\sim x$, el otro como $\sim y$.

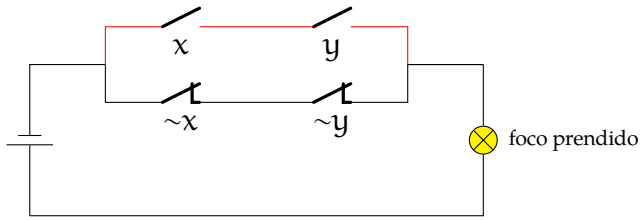


Figura 2.35 Ambos están *abiertos*, luego pasa la corriente.

Como en las operaciones anteriores, en el caso de un dispositivo de varios bits, digamos de un *byte*, la operación `XNOR` se realiza **bit a bit**, es decir,

$$\texttt{0111 0010 XNOR 0110 1011} = \texttt{1110 0110}$$

O, puesto de otra manera,

$$\begin{aligned} &\texttt{0111 0010} \\ \texttt{XNOR}\ &\texttt{0110 1011} \\ = &\texttt{1110 0110} \end{aligned}$$

Ejemplo 2.20 En un dispositivo de cuatro bits, tenemos una pareja de estados. Di cuáles son los bits que están prendidos en ambos estados o en ninguno.

$$0111, \quad 1110$$

Solución. Aplicamos la operación XNOR para ver cuáles son los bits prendidos en *ambos* estado o en *ninguno*.

$$\begin{aligned} & 0111 \\ \texttt{XNOR}\ & 1110 \\ = & 0110 \end{aligned}$$

Luego los bits prendidos en ambos estados o en ninguno son el segundo y tercer bit, es decir, el bit 1 y el 2 (de derecha a izquierda) están prendidos en *ambos* estados o están apagados en ambos. ☺

Compuerta lógica XNOR

La Figura 2.35 de la página 54 representa el circuito eléctrico de la compuerta XNOR. La salida x XNOR y es 1 si, y sólo si, *ambas* entradas, x, y, son 0, o *ambas* son 1, es decir (el símbolo $\Leftrightarrow$ se lee *si, y sólo si,*),

$$x \texttt{ XNOR } y = 1 \quad \Leftrightarrow \quad (x = 0 \ \wedge \ y = 0) \ \vee \ (x = 1 \ \wedge \ y = 1).$$

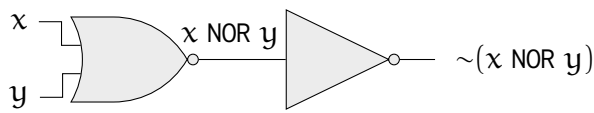


Figura 2.36 Circuito lógico de XNOR.

El dispositivo lógico que realiza la operación XNOR es la ***compuerta lógica***  .

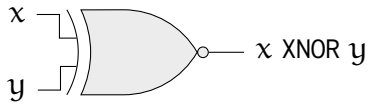


Figura 2.37 *Compuerta lógica* XNOR.

Actividad 2.9. Construye un dispositivo con una pila, cables, dos interruptores, un foco y, de ser necesario, resistencias, de manera tal que para que el foco prenda deban estar abiertos los dos interruptores o los dos cerrados. Es decir, si se cierra o abre *algún* interruptor, el foco se apaga.

Capítulo 3

Álgebra de Boole

Al terminar la lectura del presente párrafo estarás en el conjunto de las personas enteradas de que la palabra ***álgebra*** es la versión latinizada de la palabra árabe *al-jabr*, viene del libro Hidab al-jabr wa'l muqabalah, escrito por Muhammad ibn Musa al-Jwârizmî (ca. 825 d. c.).

al-Jwârizmî (de cuyo nombre viene la palabra *algoritmo*) formaba parte de la Casa de la Sabiduría (*Bayt al-Hikma*), una institución para la educación y la investigación fundada por el califa المأمون (al-Ma'mūn). En su libro, sintetizó trabajos hindúes anteriores sobre los conceptos del álgebra y usó las palabras *jabr* y *muqubalah* para designar las dos operaciones básicas para resolver ecuaciones: *jabr* significaba pasar o compensar los términos restados de un lado al otro lado de la ecuación; *muqubalah* significaba cancelar o substraer términos similares en ambos lados de la ecuación. El título de su libro se traduce como *La ciencia de compensar lo que falta y substraer los iguales*[1]

Nuestros recuerdos de la escuela secundaria asocian la palabra ***álgebra*** con las reglas de manipulación de operaciones entre polinomios, como $3x + 8$ ó $5x^2$. De seguro que una propiedad

[1] Billstein, Libeskind y Lott, *MATEMÁTICAS: Un enfoque de resolución de problemas para maestros de educación básica*, p. 196.

algebraica que recordamos es la célebre expresión del cuadrado de un binomio,

$$(x + y)^2 = x^2 + 2xy + y^2.$$

De manera similar, se emplea el vocablo *álgebra* de *conjuntos*, o de *proposiciones* o de *bits*, referida a la manipulación de las operaciones respectivas de *intersección*, *unión* y *complemento*; *conjunción*, *disyunción* y *negación*; y `AND`, `OR` y `NOT`.

El manejo de las operaciones en cada uno de esos ámbitos de conjuntos, proposiciones y bits, es similar.

George Boole presentó, de manera definitiva, en su obra de 1854, *An Investigation of the Laws of Thought, on which are founded the Mathematical Theories of Logic and Probabilities* (Una investigación de las leyes del pensamiento en las cuales se basan las teorías matemáticas de la lógica y la probabilidad), las propiedades que cumplen *variables lógicas*, caracterizadas por tener sólo los valores 0 y 1.

Posteriormente al estudio y sistematización de esas propiedades se le llamó ***álgebra de Boole***, o ***álgebra booleana***, más adelante, se designó con ese nombre a una estructura matemática abstracta.

Para nosotros, en este libro, el ***álgebra de Boole*** es el manejo de las variables a, b, c, ..., elementos de un conjunto $\mathfrak{B}$ y su comportamiento respeto a las operaciones y objetos que definiremos a continuación.

3.1. Operaciones básicas

Usamos dos operaciones binarias, la multiplicación o producto, y la suma. Cuando se diga ***multiplicación*** piensen en la *intersección* ($\cap$) o en la *conjunción* ($\wedge$, `&`, `AND`); cuando se diga ***suma***, piensen en la *unión* ($\cup$) o en la disyunción ($\vee$, `|`, `OR`).

Figura 3.1 *George Boole Monument For Lincoln 2017* de Antony Dufort.

Multiplicación o producto

Sean $a, b \in \mathfrak{B}$, denotamos su ***producto*** simplemente colocando juntas las variables, es decir, a *por* b se escribe ab, que también es un elemento de $\mathfrak{B}$, esto significa que el producto es una operación ***cerrada*** (el símbolo $\Rightarrow$ significa *implica* o *entonces*),

$$a, b \in \mathfrak{B} \;\Rightarrow\; ab \in \mathfrak{B},$$

se lee, *si* a *y* b *son elementos de* $\mathfrak{B}$, *entonces el producto* a por b (o simplemente ab) *también es elemento de* $\mathfrak{B}$.

La multiplicación cumple las siguientes propiedades básicas:

1. Idempotencia. Para todo elemento de $\mathfrak{B}$, se cumple que $aa = a$ (se usa el símbolo $\forall$ como abreviación de la expresión "*para toda*(o)"),

$$aa = a, \quad \forall a \in \mathfrak{B}.$$

2. Conmutatividad. Para cualesquiera $a, b \in \mathfrak{B}$, no importa el orden de los factores, $ab = ba$,

$$a, b \in \mathfrak{B} \;\Rightarrow\; ab = ba.$$

3. Asociatividad. La operación de multiplicación entre tres elementos puede efectuarse de varias maneras y se obtiene el mismo producto,

$$a, b, c \in \mathfrak{B} \;\Rightarrow\; (ab)c = a(bc).$$

4. Neutro multiplicativo.[2] Existe un elemento de $\mathfrak{B}$, que denotaremos con 1, que al multiplicarlo por cualquier elemento a de $\mathfrak{B}$, se obtiene a (se usa el símbolo $\exists$ para la expresión "*existe*"),

$$\exists\, 1 \in \mathfrak{B} \text{ tal que } \forall a \in \mathfrak{B},\;\; a1 = 1a = a.$$

Suma

Sean $a, b \in \mathfrak{B}$, denotamos la ***suma*** con el símbolo $+$, es decir $a + b$, que también es un elemento de $\mathfrak{B}$; esto significa que la suma es una operación ***cerrada***,

$$a, b \in \mathfrak{B} \;\Rightarrow\; a + b \in \mathfrak{B},$$

se lee, *si* a *y* b *son elementos de* $\mathfrak{B}$, *entonces la suma* a más b *también es elemento de* $\mathfrak{B}$.

La suma cumple las siguientes propiedades básicas:

1. Idempotencia. Para todo elemento de $\mathfrak{B}$, se cumple que $a + a = a$ (se usa el símbolo $\forall$ como abreviación de la expresión "*para toda*(o)"),

$$a + a = a, \quad \forall a \in \mathfrak{B}.$$

[2] Piensa en el conjunto universal Ω, $A \cap \Omega = A$.

2. Conmutatividad. Para cualesquiera $a, b \in \mathfrak{B}$, no importa el orden de los sumandos $a + b = b + a$,

$$a, b \in \mathfrak{B} \Rightarrow a + b = b + a.$$

3. Asociatividad. La operación de suma de tres elementos puede efectuarse de varias maneras y se obtiene el mismo resultado,

$$a, b, c \in \mathfrak{B} \Rightarrow (a + b) + c = a + (b + c).$$

4. Neutro aditivo.[3] Existe un elemento de $\mathfrak{B}$, que denotaremos con 0, que al sumarlo a cualquier elemento a de $\mathfrak{B}$, se obtiene a (se usa el símbolo $\exists$ para la expresión "*existe*"),

$$\exists\, 0 \in \mathfrak{B} \text{ tal que, } \forall a \in \mathfrak{B},\ a + 0 = 0 + a = a.$$

5. Suma con el Neutro multiplicativo.[4] Para cada elemento $a \in \mathfrak{B}$, se tiene que $a + 1 = 1$.

6. Producto con el Neutro aditivo.[5] Para cada elemento $a \in \mathfrak{B}$, se tiene que $a0 = 0$.

Propiedades distributivas

El producto y la suma cumplen las siguientes propiedades:

1. Producto distribuye a la suma. Cuando uno de los sumandos es un producto, como en $a + bc$, esa suma es igual al producto de la suma con cada uno de los factores, es decir $(a + b)(a + c)$,

$$a, b, c \in \mathfrak{B} \Rightarrow a + bc = (a + b)(a + c).$$

[3] Piensa en el conjunto vacío $\emptyset$, $A \cup \emptyset = A$.
[4] Piensa en la unión de un conjunto con el total Ω, $A \cup \Omega = \Omega$.
[5] Piensa en la intersección de un conjunto con el vacío $\emptyset$, $A \cap \emptyset = \emptyset$.

2. Suma distribuye al producto. Cuando en un producto, un factor es una suma, como en $a(b+c)$, ese producto es igual a la suma del producto con cada uno de los sumandos, es decir $ab + ac$,

$$a, b, c \in \mathfrak{B} \;\Rightarrow\; a(b + c) = ab + ac.$$

Nota importante. La primera propiedad, del producto distribuyendo a la suma *sólo* es válida en *álgebra booleana* o estructuras matemáticas semejantes. **No** se cumple, por ejemplo, en los números naturales $\mathbb{N}$,

$$2 + 7 \times 5 \neq (2 + 7) \times (2 + 5),$$

pues el lado izquierdo es igual a 37 mientras que el lado derecho es igual a 63.

Complemento

Dado un elemento a de $\mathfrak{B}$, le corresponde otro elemento, que llamamos ***su complemento***, y lo denotamos con un *apóstrofe*, así, el complemento de a es a' (a *prima*), cumple las siguientes propiedades:

1. La primera es que al multiplicarlo[6] por a el producto es 0,

   $$\forall a \in \mathfrak{B}, \exists\, a' \in \mathfrak{B}, \text{ tal que } a(a') = (a')a = 0.$$

   O simplemente $aa' = 0$.

2. La segunda es que al sumarle[7] a resulta 1,

   $$\forall a \in \mathfrak{B}, \exists\, a' \in \mathfrak{B}, \text{ tal que } a + (a') = (a') + a = 1.$$

   O simplemente $a + a' = 1$.

3. Ley de absorción. Complemento del complemento,

   $$\forall a \in \mathfrak{B}, \text{ se cumple } (a')' = a.$$

[6] Piensa en A^c, $A \cap A^c = \emptyset$.
[7] Piensa en A^c, $A \cup A^c = \Omega$.

Complemento y diferencia

Tanto en el ámbito de operaciones entre *bits* como entre variables booleanas, suele denotarse al complemento con $\sim a$, que ***no*** denota a la operación de *resta* o *diferencia*. Al *sumar* a con su *negativo* o *complemento*, obtenemos el *total*,

$$a + (\sim a) = 1.$$

La ***diferencia*** $a - b$ se define como

$$a - b = ab'.$$

Así, la expresión algebraica con variables booleanas $1 - x$ significa

$$\begin{aligned} 1 - x &= 1x' && \text{Definición de } \textbf{\textit{diferencia}} \\ &= x' && \text{Multiplicación por el total} \end{aligned}$$

Es decir, $x' = \sim x = 1 - x$. *La expresión* $-x$ ***no*** *tiene sentido*.

Suma excluyente

Usamos el complemento y las operaciones básicas de suma y producto para definir otra operación: es semejante a la *diferencia simétrica* en conjuntos o a `XOR` en circuitos, la denotamos con $\oplus$ y se define como

$$a \oplus b = ab' + ba'.$$

Orden parcial

Entre las *variables booleanas* hay una ***relación*** de ***orden parcial***[8] que denotamos con $\preccurlyeq$, decimos que a ***precede*** o ***antecede*** b, si $ab = a$,

$$a, b \in \mathfrak{B};\ \ ab = a \ \Leftrightarrow\ a \preccurlyeq b,$$

[8] Semejante a la contención en conjuntos

se lee *si* a *y* b *son elementos de* $\mathfrak{B}$ *entonces* $ab = a$ *si, y sólo si,* a ***precede*** b. También se dice que b ***sucede*** o es ***un sucesor*** de a.

La condición *si, y sólo si* en la descripción de ***predecesor*** significa que si $ab = a$ entonces $a \preccurlyeq b$ y viceversa, que si $a \preccurlyeq b$, entonces $ab = a$.

La *relación* de *predecesor* cumple las propiedades:

1. Reflexiva. Para cada a en $\mathfrak{B}$, a precede a,
$$\forall a \in \mathfrak{B}, \quad a \preccurlyeq a.$$

2. Antisimétrica (Criterio de igualdad). Para cada a y b en $\mathfrak{B}$, tales que a precede b, si sucede que además b precede a, entonces $a = b$,
$$a, b \in \mathfrak{B}; \quad a \preccurlyeq b \ \text{ y } \ b \preccurlyeq a \ \Rightarrow \ a = b.$$

3. Transitiva. Si a precede b y b precede c, entonces a precede c,
$$a, b, c \in \mathfrak{B}; \quad a \preccurlyeq b \ \text{ y } \ b \preccurlyeq c \ \Rightarrow \ a \preccurlyeq c.$$

Nota importante. Dados dos elementos cualesquiera de $\mathfrak{B}$, no necesariamente son ***comparables***, es decir, si d y $f \in \mathfrak{B}$ no necesariamente se tiene que
$$d \preccurlyeq f \ \text{ o } \ f \preccurlyeq d.$$
En ese caso, los elementos d y f ***no*** son *comparables*[9]

[9] Como en el caso de la contención de conjuntos, no necesariamente $A \subseteq B$ o $B \subseteq A$.

Criterio de igualdad

La propiedad *antisimétrica* de la precedencia nos dió un criterio de igualdad. Los elementos a y $b \in \mathfrak{B}$ son ***iguales*** si, y sólo si, $a \preccurlyeq b$ y $b \preccurlyeq a$. Expresado en términos de *producto*,

$$a, b \in \mathfrak{B};\ \ a = b \ \Leftrightarrow\ ab = a \ \text{ y } \ ba = b.$$

se lee: *sean* a, b *dos elementos de* $\mathfrak{B}$, a *es igual a* b *si, y sólo si,* a por b *es igual a* a ***y*** b *por* a *es igual a* b.

Esta *relación* de igualdad cumple con:

1. Reflexiva. Para cada a en $\mathfrak{B}$, a es igual a a,

$$\forall a \in \mathfrak{B},\ \ a = a.$$

2. Simétrica. Si a y b son elementos de $\mathfrak{B}$ y a es igual a b, entonces b es igual a a,

$$\forall a, b \in \mathfrak{B} \ \text{ tales que } \ a = b, \ \text{ entonces } \ b = a.$$

3. Transitiva. Si a, b y c son elementos de $\mathfrak{B}$ tales que a es igual a b y b es igual a c, entonces a es igual a c,

$$\forall a, b, c \in \mathfrak{B} \ \text{ tales que } \ a = b \ \text{ y } \ b = c, \ \text{ entonces } \ a = c.$$

Nota importante. Cualquier *relación* que cumpla las tres propiedades anteriores, de ser: *reflexiva*, *simétrica* y *transitiva*, se llama una ***relación de equivalencia***. Así, la igualdad en álgebra de Boole, es una relación de equivalencia.

Un caso particular de *precedencia*, es la ***precedencia estricta***, se denota con $\prec$. Para que a ***preceda estrictamente*** b, se debe tener que a preceda b, pero que a no sea igual a b, así

$$a \prec b \ \Leftrightarrow\ ab = a \ \text{ y } \ a \neq b.$$

Esta relación de *precedencia estricta* **no** es *reflexiva*, pues aunque $a \preccurlyeq a$, no es cierto que $a \neq a$; pero **sí** es *transitiva*, es decir

$$a \prec b \ \text{ y } \ b \prec c \ \Rightarrow\ a \prec c.$$

3.2. Leyes de De Morgan

Una vez definidas las operaciones básicas del álgebra booleana, una manera fundamental de combinar suma y producto con el complemento la expone, en su forma moderna, Augustus De Morgan en su obra *Formal Logic or The Calculus of Inference, Necessary and Probable*, p. 118, en la Figura 3.2 vemos un extracto.

I may now proceed to extend this idea and notation relative to propoſitions of complex terms. The complexity conſiſts in the terms being conjunctively or disjunctively formed from other terms, as in PQ, that to which both the names P and Q belong conjunctively; and as in P,Q that to which one (or both) of the names P and Q belong disjunctively. The contrary of PQ is p,q; that of P,Q is pq. *Not both* is either not one or not the other, or not either. *Not either P nor Q* (which we might denote by :P,Q or .P,Q) is logically '*not* P *and not* Q' or pq: and this is then the contrary of P,Q.

Figura 3.2 No ambos es no uno *o* no otro. No, P o Q, es no P *y* no Q.

Leyes de De Morgan. Si a y b son elementos de $\mathfrak{B}$, se cumple:

1. El complemento del *producto* es igual a la *suma* de los complementos
$$(ab)' = a' + b'.$$
Esta propiedad es la operación `NAND` en circuitos.

2. El complemento de la *suma* es igual al *producto* de los complementos
$$(a + b)' = a'b'.$$
Esta propiedad es la operación `NOR` en circuitos.

Usamos estas leyes de De Morgan junto con las propiedades de las operaciones básicas para operar *expresiones booleanas*

Ejemplo 3.1 La suma excluyente se puede expresar como

$$a \oplus b = (a + b)(ab)'.$$

Solución. Para ello, desarrollamos el lado derecho mediante las propiedades de las operaciones,

$$\begin{aligned}
(a+b)(ab)' &= (a+b)(a'+b') && \text{Ley de De Morgan}\\
&= (a+b)a' + (a+b)b' && \text{Distributividad}\\
&= aa' + ba' + ab' + bb' && \text{Distributividad}\\
&= 0 + ba' + ab' + 0\\
&= ab' + ba'\\
&= a \oplus b
\end{aligned}$$

☺

Principio de *Dualidad*. Para obtener el complemento de alguna expresión, debido a las leyes de De Morgan, basta *reemplazar* cada variable por su complemento e *intercambiar* las operaciones de producto y suma, respetando los factores (sumandos) de cada operación.

Ejemplo 3.2 Si aplicamos el ***principio de dualidad*** a la expresión $a+b(d+ca')$, obtenemos su complemento $a'\left[b'+d'(c'+a)\right]$. Verifica.

Solución. Aplicamos sucesivamente las leyes de De Morgan:

$$\begin{aligned}
\left[a+b(d+ca')\right]' &= a'\left[b(d+ca')\right]'\\
&= a'\left[b'+(d+ca')'\right]\\
&= a'\left[b'+d'(ca')'\right]\\
&= a'\left[b'+d'(c'+a)\right].
\end{aligned}$$

☺

Problema 3.1 Simplifica las siguientes expresiones booleanas por medio del *principio de dualidad* y verifica tu respuesta aplicando las leyes de De Morgan cada vez que sea necesario

1. $(ab' + c)'$ 2. $\left[ad(b' + c)\right]'$

3.3. Algunas propiedades

A partir de las propiedades con que definimos las operaciones, podemos *operar* variables boolenas para simplificar expresiones y verificar igualdades.

En su tesis de 1936, publicada en 1938, «A Symbolic Analysis of Relay and Switching Circuits», Shannon señala, entre otras, cuatro propiedades que mencionamos aquí como ejemplos.

En los Ejemplos siguientes, sean a, b y $c \in \mathfrak{B}$, se tiene que:

Ejemplo 3.3 $a + ab = a$.

Solución. Desarrollando la expresión del lado izquierdo,

$$\begin{aligned} a + ab &= a1 + ab && \text{Neutro multiplicativo} \\ &= a(1 + b) && \text{Distributividad} \\ &= a1 && 1 + b = 1 \\ &= a && a1 = a \end{aligned}$$

Ejemplo 3.4 $a(a + b) = a$.

Solución. Partimos del lado izquierdo

$$\begin{aligned} a(a + b) &= aa + ab && \text{Distributividad} \\ &= a + ab && \text{Idempotencia} \\ &= a && \text{Ejemplo anterior} \end{aligned}$$

Ejemplo 3.5 $ab + a'c = ab + a'c + bc$

Solución. Partimos del lado izquierdo, por el Ejemplo 3.3 tenemos que $b = b + bc$ y $c = c + bc$, luego

$$\begin{aligned} ab + a'c &= a(b + bc) + a'(c + bc) \\ &= ab + abc + a'c + a'bc && \text{Distributividad} \\ &= ab + a'c + abc + a'bc && \text{Asociatividad} \\ &= ab + a'c + (a + a')bc && \text{Distributividad} \\ &= ab + a'c + (1)bc && a + a' = 1 \\ &= ab + a'c + bc \end{aligned}$$

☺

Ejemplo 3.6 $(a + b)(a' + c) = (a + b)(a' + c)(b + c)$

Solución. Comenzamos desarrollando el lado izquierdo,

$$\begin{aligned} (a + b)(a' + c) &= (a + b)a' + (a + b)c && \text{Distributividad} \\ &= aa' + ba' + ac + bc && \text{Distributividad} \\ &= 0 + ba' + ac + bc && aa' = 0 \\ &= ac + a'b + bc && \text{Neutro aditivo} \end{aligned}$$

Ahora desarrollamos el lado derecho; nota que los dos primeros factores del lado derecho son iguales al lado izquierdo, recién calculado; así, tenemos

$$\begin{aligned} (a + b)(a' + c)(b + c) &= (ac + a'b + bc)(b + c) \\ &= abc + a'bb + bbc + acc + a'bc + bcc \\ &= abc + a'b + bc + ac + a'bc + bc \\ &= ac + (a + a')bc + a'b + bc + bc \\ &= ac + 1bc + a'b + bc \\ &= ac + a'b + bc \end{aligned}$$

Desarrollamos ambos lados y llegamos al mismo resultado, luego la igualdad es cierta. ☺

Hay otras propiedades de apariencia sencilla pero que facilitan la simplificación de expresiones booleanas algebraicas,

Ejemplo 3.7 Demuestra que $a(a' + b) = ab$.

Solución.

$$\begin{aligned} a(a' + b) &= aa' + ab && \text{Distributividad} \\ &= 0 + ab && aa' = 0 \\ &= ab \end{aligned}$$

Ejemplo 3.8 Demuestra que $a + a'b = a + b$.

Solución. Partimos del lado izquierdo,

$$\begin{aligned} a + a'b &= (a + a')(a + b) && \text{Distributividad} \\ &= 1(a + b) && a + a' = 1 \\ &= a + b && 1(a + b) = a + b \end{aligned}$$

Ejemplo 3.9 Demuestra que $a'(a + b) = a'b$.

Solución.

$$\begin{aligned} a'(a + b) &= a'a + a'b && \text{Distributividad} \\ &= 0 + a'b && a'a = 0 \\ &= a'b \end{aligned}$$

Ejemplo 3.10 Demuestra que $a' + ab = a' + b$.

Solución.

$$\begin{aligned} a' + ab &= (a' + a)(a' + b) && \text{Distributividad} \\ &= 1(a' + b) && a' + a = 1 \\ &= a' + b \end{aligned}$$

Los dos ejemplos anteriores se pueden deducir de los dos que les anteceden con a' en lugar de a.

Capítulo 4

Álgebra de circuitos

En el capítulo anterior vimos algunas de las propiedades algebraicas de las variables booleanas (su valor es 0 ó 1) desarrolladas por Boole en su obra *An Investigation of the Laws of Thought, on which are founded the Mathematical Theories of Logic and Probabilities* (Una investigación de las leyes del pensamiento en las cuales están fundamentadas las teorías matemáticas de la lógica y las probabilidades) y retomadas posteriormente por Shannon en su famosa tesis «A Symbolic Analysis of Relay and Switching Circuits» (Un análisis simbólico de conexiones y circuitos de conmutacion).

Shannon emplea las más diversas expresiones algebraicas booleanas para representar un circuito eléctrico y transforma las expresiones por medio las operaciones algebraicas booleanas de las secciones anteriores, y obtiene circuitos equivalentes al original.

4.1. Simplificación de circuitos

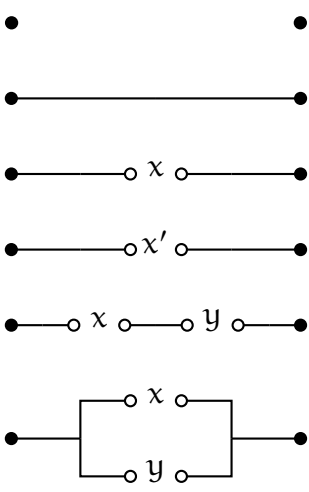
Como menciona Shannon en su artículo «The Synthesis of Two Terminal Switching Circuits» (La síntesis de circuitos de dos

terminales), la *teoría de circuitos* se puede dividir en dos partes principales, el *análisis* y la *síntesis*. El problema de análisis consiste en determinar la manera en que debe operar un circuito dado; el problema inverso de hallar un circuito que satisfaga ciertas condiciones dadas y, en particular, el *mejor* circuito, es un problema más difícil y más importante desde el punto de vista práctico.

Mencionamos como dato histórico, que actualmente se usa una notación *dual* a la empleada por Shanon, quien empleaba la terminología $a + b$ para representar un circuito *en serie* y ab para uno en *paralelo*.

Representamos una variable booleana en un circuito como  , una recta con una conexión que puede estar *abierta* o *cerrada*, donde $x = 0$ significa que está abierta y $x = 1$ significa que está cerrada,

Combinamos variables por medio de las operaciones de suma y producto las representamos por medio de conexiones y obtenemos nuevos circuitos, como vemos en la figura 4.1



0 Conexión abierta permanente.

1 Conexión cerrada permanente.

x Hacer contacto en x.

x' Impedir contacto en x.

xy Conexión en serie.

$x + y$ Conexión en paralelo.

Figura 4.1 Diagramas en red de circuitos

Ahora podemos usar el álgebra de Boole para simplificar circuitos.

Ejemplo 4.1 Simplica el circuito de la figura siguiente (4.2)

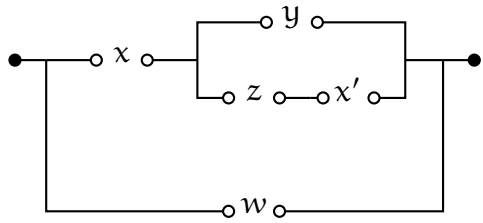


Figura 4.2 Simplifica $w + x\left[y + (zx')\right]$.

Solución. El circuito de la figura corresponde a la expresión: $w + x\left[y + (zx')\right]$. Aplicamos las propiedades algebraicas

$$\begin{aligned} w + x\left[y + (zx')\right] &= w + xy + x(zx') && \text{Distributividad} \\ &= w + xy + (xx')z && \text{Asociatividad} \\ &= w + xy + 0z && xx' = 0 \\ &= w + xy + 0 && 0z = 0 \\ &= w + xy && \text{Neutro aditivo} \end{aligned}$$

Tenemos entonces que

$$w + x\left[y + (zx')\right] = w + xy,$$

el diagrama correspondiente a $w + xy$ es

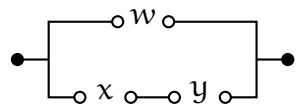


Figura 4.3 $w + x\left[y + (zx')\right] = w + xy$.

Los diagramas de las figuras 4.2 y 4.3 son equivalentes. Aquí lo mostramos mediante álgebra de Boole, otra manera de hacerlo sería verificar que sus tablas de verdad son iguales. ☺

Veamos otro ejemplo.

Ejemplo 4.2 Simplifica el circuito

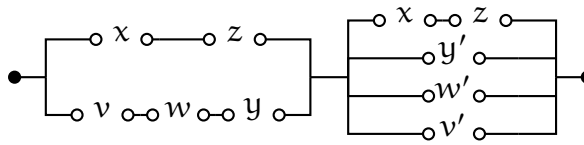


Figura 4.4 Hallar la expresión algebraica y simplificar.

Solución. Como lo sugiere el pie de la figura 4.4, primero hallemos la expresión algebraica del circuito, ésta es:

$$(xz + vwy)(xz + y' + w' + v').$$

Ahora simplifiquemos la expresión algebraica mediante las propiedades que vimos en el Capítulo 3, en particular los ejemplos 3.4 y 3.3, ¿puedes identificar cuándo se usan?

$$\begin{aligned}(xz + vwy)(xz + y' + w' + v') &= xz(xz + y' + w' + v') + \\ &\quad vwy(xz + y' + w' + v') \\ &= xz + vwy(xz + y' + w' + v') \\ &= xz + vwyxz + vwyy' + \\ &\quad vwyw' + vwyv' \\ &= xz + vwyxz \\ &= xz,\end{aligned}$$

el diagrama correspondiente es,

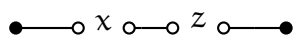


Figura 4.5 Circuito xz.

Los diagramas de las figuras 4.4 y 4.5 son equivalentes pues lo son las expresiones:

$$(xz + vwy)(xz + y' + w' + v') = xz.$$

Problema 4.1 Este problema es similar al propuesto por Barnett en su interesante proyecto *Applications of Boolean Algebra: Claude Shannon and Circuit Design*, p. 10 (Aplicaciones del álgebra booleana: Claude Shannon y el diseño de circuitos).

Traza el diagrama correspondiente a la expresión

$$w(xy + w')(s'yz' + vyz' + swz + xz),$$

simplifícalo y muestra la expresión equivalente con su diagrama.

*4.2. Simplificación de funciones booleanas

Esta sección marcada con un *asterisco* *, es optativa pues usa el concepto de ***función***, que no hemos estudiado[1]. Daremos una muy breve explicación y presentaremos varias fórmulas muy útiles para simplificar ***funciones booleanas***.

Ya mencionamos que las ***variables booleanas*** son símbolos x, y, z, w, x_1, x_2, x_3, ..., que pueden tomar los valores 0 ó 1 y se les aplican las operaciones ya mencionadas.

Dicho de manera sencilla, una función booleana es una expresión algebraica formada por variables booleanas combinadas con las operaciones estudiadas en el Capítulo 3. Por ejemplo,

$$f(x, y, z) = x'z' + xyz + yz' + x'y'z + x'yz.$$

La idea del concepto de *función* es que al asignar valores (en este caso, 0 ó 1) a las variables x, y y z, y efectuar las operaciones, se obtiene el ***valor*** de la función. Digamos que asignamos los

[1] Ver la definición de ***función*** en *Conjuntos, lógica y funciones*, p. 151.

valores $x = 0$, $y = 1$, $z = 1$, entonces podemos *valuar* la función en el *punto* $(x, y, z) = (0, 1, 1)$ y obtenemos

$$\begin{aligned} f(0,1,1) &= 0' \cdot 1' + 0 \cdot 1 \cdot 1 + 1 \cdot 1' + 0' \cdot 1' \cdot 1 + 0' \cdot 1 \cdot 1 \\ &= 1 \cdot 0 + 0 \cdot 1 \cdot 1 + 1 \cdot 0 + 1 \cdot 0 \cdot 1 + 1 \cdot 1 \cdot 1 \\ &= 0 + 0 + 0 + 0 + 1 \\ &= 1. \end{aligned}$$

En lugar de efectuar todas esas operaciones, buscamos *simplificar la función*, simplificando su expresión algebraica, en este caso,

$$\begin{aligned} f(x,y,z) &= x'z' + xyz + yz' + x'y'z + x'yz \\ &= x'z' + yz(x + x') + yz' + x'z(y' + y) \\ &= x'z' + yz + yz' + x'z \\ &= x'(z + z') + y(z + z') \\ &= x' + y. \end{aligned}$$

Obtenemos así, que $f(x, y, z) = x' + y$, y para calcular el valor de f en el punto $(0, 1, 1)$, substituimos,

$$f(0,1,1) = 0' + 1 = 1 + 1 = 1,$$

lo cual coincide con la valuación en la expresión sin simplificar.

Problema 4.2 Explica cómo se obtuvo la igualdad

$$x'z' + xyz + yz' + x'y'z + x'yz = x' + y$$

en el desarrollo anterior.

Descomposición de Boole

En la ya varias veces citada obra, «*An Investigation of the Laws of Thought*...», Boole propone [p. 72] la siguiente descomposición de una función (ahora llamada *booleana*) de una variable *lógica*, es decir, que sólo toma alguno de los valores 0 ó 1; plantea expresar una función como combinación de la variable y su complemento, es decir

$$f(x) = ax + bx', \qquad (*)$$

Para ello, calcula primero $f(1)$,

$$\begin{aligned} f(1) &= a \cdot 1 + b \cdot 1' \\ &= a \cdot 1 + b \cdot 0 && 1' = 0 \\ &= a + 0 \\ &= a. \end{aligned}$$

Después calcula $f(0)$,

$$\begin{aligned} f(0) &= a \cdot 0 + b \cdot 0' \\ &= a \cdot 0 + b \cdot 1 && 0' = 1 \\ &= 0 + b \\ &= b. \end{aligned}$$

Obtenemos así los valores $a = f(1)$ y $b = f(0)$. Los substituimos en la ecuación $(*)$,

$$\begin{aligned} f(x) &= ax + bx' \\ &= f(1)x + f(0)x', \end{aligned}$$

y obtenemos la siguiente fórmula de descomposición de una función en la variable y su complemento,

$$f(x) = f(1)x + f(0)x'. \qquad (**)$$

Ejemplo 4.3 Expresa la función $f(x) = (1-x)x' + (1+x')x$ como combinación de x y x'.

Solución. Valuamos $f(1)$ y $f(0)$.
$f(1) = (1-1)0 + (1+0)1 = 1$,
$f(0) = (1-0)1 + (1+1)1 = 1$.
Por lo tanto $f(x) = x + x'$. ☺

Problema 4.3 Simplifica la función del ejemplo 4.3 anterior, por medio de operaciones algebraicas booleanas, para verificar el resultado.

Después Boole extiende la fórmula (**) para varias variables, trabajo en el que se basa Shannon.

Fórmulas de Shannon

Cómo ya mencionamos al final de la subsección anterior, Shannon usa la ampliación a varias variables de la descomposición de Boole y establece unas fórmulas que son muy útiles para simplificar funciones que, en este caso, representan circuitos.

Las fórmulas son conocidas y referidas según la numeración que tienen en el artículo «A Symbolic Analysis of Relay and Switching Circuits».

La primera, la (**10a**), es la generalización de la fórmula (**) de la página 77,

$$f(x_1, x_2, \dots, x_n) = x_1 \cdot f(1, x_2, \dots, x_n) + x_1' \cdot f(0, x_2, \dots, x_n), \tag{10a}$$

$$f(x_1, x_2, \dots, x_n) = \left[f(0, x_2, \dots, x_n) + x_1\right] \cdot \left[f(1, x_2, \dots, x_n) + x_1'\right]. \tag{10b}$$

En la fórmula (**10a**) la función se descompone como *suma* o *conjunción*, mientras que en la fórmula (**10b**), la función se descompone como *producto* o *disyunción*.

Las fórmulas a continuación son particularmente útiles para simplificar expresiones algebraicas booleanas,

$$xf(x, y, z, \dots) = xf(1, y, z, \dots), \tag{17a}$$
$$x + f(x, y, z, \dots) = x + f(0, y, z, \dots). \tag{17b}$$

$$x'f(x, y, z, \dots) = x'f(0, y, z, \dots), \tag{18a}$$
$$x' + f(x, y, z, \dots) = x' + f(1, y, z, \dots). \tag{18b}$$

Las fórmulas (**10a**) y (**10b**), así como las (**17a**), (**17b**), (**18a**) y (**18b**) se demuestran por lo que Shannon llama ***inducción perfecta***, que consiste en substituir la variable en cuestión, por ejemplo x_1 en el caso de (**10a**) ó x en el caso de (**17b**), por los valores 0 y 1 y verificar que en ambos casos se obtiene una identidad.

Ejemplo 4.4 Demuestra por el método de *inducción perfecta* de Shannon la fórmula (**10a**) de la página 78.

Solución. Para demostrar la fórmula por el método de *inducción perfecta*, debemos verificar que al substituir la variable en cuestión por 0 y después por 1, en ambos casos obtenemos una identidad.

Substituimos $x_1 = 0$ en

$$f(x_1, x_2, \dots, x_n) = x_1 \cdot f(1, x_2, \dots, x_n) + x_1' \cdot f(0, x_2, \dots, x_n).$$

Del lado izquierdo obtenemos $f(0, x_2, \dots, x_n)$, y del lado derecho,

$$\begin{aligned} 0 \cdot f(1, x_2, \dots, x_n) + 0' \cdot f(0, x_2, \dots, x_n) &= 1 \cdot f(0, x_2, \dots, x_n) \\ &= f(0, x_2, \dots, x_n). \end{aligned}$$

Verificamos que es una identidad.

Ahora substituimos $x_1 = 1$. Del lado izquierdo obtenemos $f(1, x_2, \dots, x_n)$, y del lado derecho,

$$\begin{aligned} 1 \cdot f(1, x_2, \dots, x_n) + 1' \cdot f(0, x_2, \dots, x_n) &= 1 \cdot f(1, x_2, \dots, x_n) \\ &= f(1, x_2, \dots, x_n). \end{aligned}$$

Verificamos de nuevo que es una identidad.

Así, la fórmula queda demostrada. ☺

Problema 4.4 Demuestra, por el método de *inducción perfecta*, la fórmula (**18b**) de la página 79.

Veamos las fórmulas en acción, aplicándolas a un ejemplo del artículo de Shannon.

Ejemplo 4.5 Simplifica el circuito siguiente,

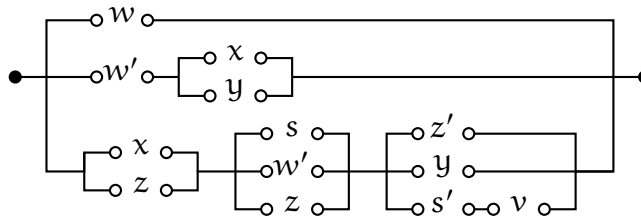


Figura 4.6 Simplifica la función que lo representa.

La función que representa al circuito es

$$f(s, v, w, x, y, z) = w + w'(x+y) + (x+z)(s+w'+z)(z'+y+s'v),$$

que podemos expresar como

$$f(s, v, w, x, y, z) = w + g(s, v, w, x, y, z),$$

donde $g(s, v, w, x, y, z) = w'(x + y) + (x + z)(s + w' + z)(z' + y + s'v)$; aplicamos la fórmula (**17b**), simplemente reordenando las variables, así

$$\begin{aligned} f(s,v,w,x,y,z) &= w + g(s,v,w,x,y,z) \\ &= w + g(s,v,0,x,y,z) \\ &= w + \big[0'(x+y)+ \\ &\qquad (x+z)(s+0'+z)(z'+y+s'v)\big] \\ &= w + x + y + (x+z)(s+1+z)(z'+y+s'v) \\ &= w + x + y + (x+z)(z'+y+s'v). \end{aligned}$$

(En el último paso *eliminamos* el término $(s+1+z)$, ¿por qué?)

Ahora $f(s,v,w,x,y,z) = w + x + y + (x+z)(z'+y+s'v)$, la expresamos como

$$f(s,v,w,x,y,z) = x + h(s,v,w,x,y,z)$$

donde $h(s,v,w,x,y,z) = w+y+(x+z)(z'+y+s'v)$. Aplicamos ahora a x la fórmula (**17b**),

$$\begin{aligned} f(s,v,w,x,y,z) &= x + h(s,v,w,x,y,z) \\ &= x + h(s,v,w,0,y,z) \\ &= x + w + y + (0+z)(z'+y+s'v) \\ &= w + x + y + z(z'+y+s'v). \end{aligned}$$

Ahora $f(s,v,w,x,y,z) = w + x + y + z(z'+y+s'v)$, la expresamos como

$$f(s,v,w,x,y,z) = y + k(s,v,w,x,y,z)$$

donde $k(s,v,w,x,y,z) = w + x + z(z'+y+s'v)$. Aplicamos ahora a y la fórmula (**17b**),

$$\begin{aligned} f(s,v,w,x,y,z) &= y + k(s,v,w,x,y,z) \\ &= y + k(s,v,w,x,0,z) \\ &= y + w + x + z(z'+0+s'v) \\ &= w + x + y + z(z'+s'v). \end{aligned}$$

Mediante la aplicación sucesiva de la fórmula (**17b**) a diferentes variables hemos reducido la función a la expresión

$$f(s, v, w, x, y, z) = w + x + y + z(z' + s'v).$$

Al efectuar el producto indicado en el último sumando obtenemos zz', que es igual a 0, así, la función se simplifica,

$$f(s, v, w, x, y, z) = w + x + y + zs'v.$$

El circuito propuesto se simplifica como lo indica el siguiente diagrama,

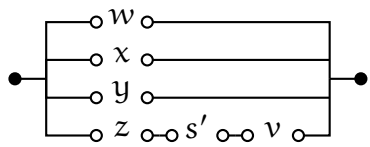
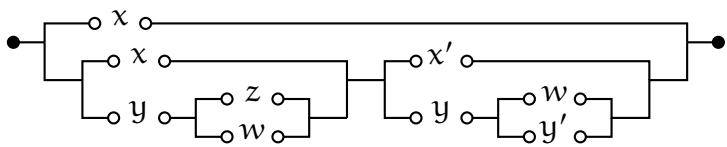


Figura 4.7 Circuito simplificado.

Problema 4.5 Simplifica el circuito siguiente



4.3. *Wolfram|Alpha*

Es importante desarrollar la habilidad de realizar operaciones booleanas, o de lógica proposicional.

Así como realizamos con facilidad las diarias operaciones de suma, resta, multiplicación y división, así debemos manipular

las operaciones de negación, conjunción y disyunción; la mejor manera desarrollar esta habilidad, es practicar.

Sin embargo, al resolver problemas complicados, en lugar de realizar las operaciones *a mano*, empleamos una calculadora. El punto importante al emplear una calculadora para resolver problemas aritméticos, es elegir las operaciones adecuadas, proporcionar los datos de manera correcta y saber interpretar el resultado, así como percibir si el resultado mostrado por la calculadora es *razonable*.

Hay medios, como *Wolfram|Alpha* en Internet, `https://www.wolframalpha.com/`, para simplificar expresiones lógicas. Es de fácil uso, hay un renglón dónde se coloca la expresión lógica usando las palabras `not`, `and`, `or`, para las operaciones de *negación*, *disyunción* y *conjunción*, respectivamente. Por ejemplo la expresión $(p \vee q)'$ la colocamos como "`not(p or q)`" (se omiten las *comillas*) y oprimimos `<Enter>`.

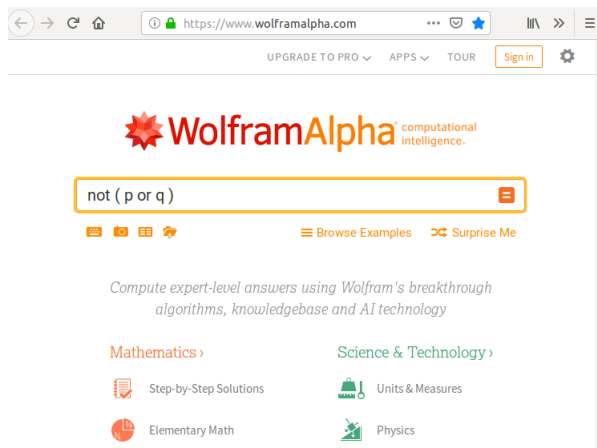


Figura 4.8 Coloca la expresión y oprime `<Enter>` o pícale al signo de *igual* al final del renglón.

Se presenta en pantalla la expresión colocada detectada, en este caso $\neg(p \vee q)$, su tabla de verdad y varias expresiones equivalentes.

Figura 4.9 Tabla de verdad y formas mínimas equivalentes.

En el renglón señalado como `DNF` (*Disjuntive Normal Form*, Forma normal disjuntiva), vemos que la expresión colocada es equivalente a $\neg p \wedge \neg q$; ley de De Morgan, página 45.

Ejemplo 4.6 Simplifica el circuito

$$w(xy + w')(s'yz' + vyz' + swz + xz).$$

Solución. Colocamos en *Wolfram|Alpha* la expresión

```
w and ((x and y) or not w) and ((not s and y and
not z) or (v and y and not z) or (s and w and
z) or (x and z))
```

En la Figura 4.10 de la página 85 vemos que reconoce la expresión colocada y la escribe en términos de las operaciones $\neg$, $\wedge$ y $\vee$. También presenta varias expresiones equivalentes.

Escogemos una simple, la forma normal conjuntiva (CNF), $(\neg s \vee v \vee z) \wedge w \wedge x \wedge y$ y tenemos finalmente que

$$w(xy + w')(s'yz' + vyz' + swz + xz) = wxy(s' + v + z).$$

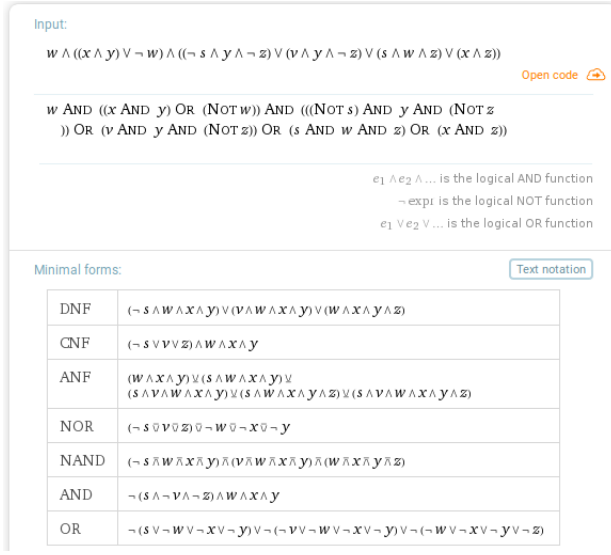


Figura 4.10 Reconoce la expresión y presenta formas mínimas equivalentes.

Problema 4.6 Compara el resultado obtenido en el ejemplo 4.6 con el del Problema 4.1 al simplificar el circuito

$$w(xy + w')(s'yz' + vyz' + swz + xz).$$

Solución a los problemas

Solución 1.1 Nos piden formar atuendos de tres prendas, claramente se dividen en lo que podemos usar en la cabeza, en el cuerpo y de calzado. Hay 3 prendas para la cabeza (el casco, las trenzas y la máscara), hay 4 prendas para el cuerpo (el *kilt*, el mameluco, el tutú y la bata) y hay 2 prendas para el calzado (los huaraches y las botas). Aplicamos el Principio fundamental del conteo, luego hay $3 \times 2 \times 4 = 24$ atuendos diferentes que podemos formar con esas prendas.

Solución 1.2 Denotemos con Ω el conjunto de estados de un dispositivo de cuatro bits,

$$\begin{aligned}\Omega = \{\, &\texttt{0000}, \texttt{0001}, \texttt{0010}, \texttt{0011},\\ &\texttt{0100}, \texttt{0101}, \texttt{0110}, \texttt{0111},\\ &\texttt{1000}, \texttt{1001}, \texttt{1010}, \texttt{1011},\\ &\texttt{1100}, \texttt{1101}, \texttt{1110}, \texttt{1111} \,\}\end{aligned}$$

El conjunto A con el primer, tercer y cuarto bit prendido (bits `0`, `2` y `3`) es

$$A = \{\, \texttt{1011}, \texttt{1111} \,\}$$

y el conjunto B, tercer y cuarto bit prendidos,

$$B = \{\, \texttt{0011}, \texttt{0111}, \texttt{1011}, \texttt{1111} \,\},$$

Vemos que cada elemento de A es un elemento de B, luego $A \subseteq B$.

Solución 1.3 Vemos que son 6 platillos, constan de base y aderezo, el de *queso parmesano* puede ir con las dos bases, *pasta* y *ensalada*, luego

$$Q = \{\text{pasta con queso parmesano, ensalada con queso parmesano}\}.$$

El complemento de Q es el conjunto de los platillos que ***no*** lleven queso parmesano,

$$Q^c = \{\text{pasta con aceite de oliva, pasta con aceitunas, ensalada con aceite de oliva, ensalada con aceitunas}\}$$

Según vemos en la descripción de los platillos, ninguno lleva salsa catsup, luego $R = \emptyset$.

Solución 1.4

1. La negación de "El número 5 es par" es "El número 5 **no** es par", lo cual es equivalente a decir que "El número 5 es impar". Esto último es verdadero, así que el *valor de verdad* de la negación es *Verdadero*.

2. Ahora todos sabemos que es cierto que "La Tierra gira alrededor del Sol". La negación de lo anterior, "La Tierra **no** gira alrededor del Sol", es *Falso*.

3. Es frecuente conocer casos de personas envenenadas por comer hongos, luego *sí* es cierto (es *Verdadero*) que "Hay hongos venenosos", la negación es "**No** hay hongos venenosos", lo cual es *Falso*.

Recuerda que si una proposición es Verdadera, su negación es Falsa, y *viceversa*.

Solución 1.5

1. Por el *Principio general del conteo*, el conjunto Ω de los estados de un dispositivo de cuatro bits tiene $2^4 = 16$ elementos que

listamos a continuación

$$\begin{aligned}\Omega = \{\, &\texttt{0000}, \texttt{0001}, \texttt{0010}, \texttt{0011},\\ &\texttt{0100}, \texttt{0101}, \texttt{0110}, \texttt{0111},\\ &\texttt{1000}, \texttt{1001}, \texttt{1010}, \texttt{1011},\\ &\texttt{1100}, \texttt{1101}, \texttt{1110}, \texttt{1111} \,\}\end{aligned}$$

2. El conjunto de los estados que tienen prendido el primer y tercer bit es

$$A = \{\, \texttt{0101}, \texttt{0111}, \texttt{1101}, \texttt{1111} \,\}$$

El complemento del conjunto anterior es

$$\begin{aligned}A^c = \{\, &\texttt{0000}, \texttt{0001}, \texttt{0010}, \texttt{0011},\\ &\texttt{0100}, \texttt{0110}, \texttt{1000}, \texttt{1001},\\ &\texttt{1010}, \texttt{1011}, \texttt{1100}, \texttt{1110} \,\}\end{aligned}$$

3. El *negativo* de cada estado en A es

$$\begin{aligned}\sim\texttt{0101} &= \texttt{1010}\\ \sim\texttt{0111} &= \texttt{1000}\\ \sim\texttt{1101} &= \texttt{0010}\\ \sim\texttt{1111} &= \texttt{0000}\end{aligned}$$

Así,

$$B = \{\, \texttt{1010}, \texttt{1000}, \texttt{0010}, \texttt{0000} \,\}.$$

Vemos que $B \neq A^c$. En B simplemente consideramos los estados obtenidos de prender lo apagado y apagar lo prendido en los elementos de A, mientras que en A^c se incluyen *todos* los estados que no tienen prendido el bit 0 **o** el bit 2.

Solución 2.1 Construimos la tabla de verdad de $\sim(x \,\&\, y)$ añadiendo esa columna a la tabla de $x \,\&\, y$.

x	y	$x \,\&\, y$	$\sim(x \,\&\, y)$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

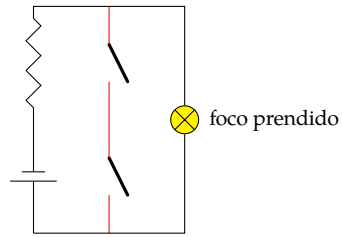
Solución 2.2 Construimos la tabla de verdad de $\sim(x \mid y)$ añadiendo esa columna a la tabla de $x \mid y$.

x	y	$x \mid y$	$\sim(x \mid y)$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Solución 2.3 La tabla de verdad de NAND es

x	y	x **NAND** y
0	0	1
0	1	1
1	0	1
1	1	0

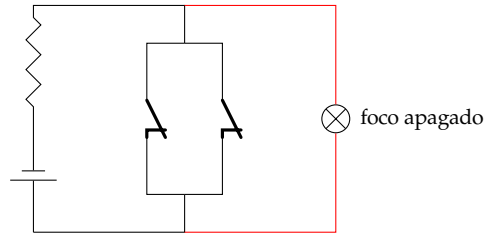
En el primer renglón vemos que si $x = y = 0$, entonces x NAND $y = 1$, luego el foco está prendido. El diagrama del circuito es



Solución 2.4 La tabla de verdad de NOR es

x	y	x NOR y
0	0	1
0	1	0
1	0	0
1	1	0

En el último renglón vemos que si $x = y = 1$, entonces x NOR $y = 0$, luego el foco está apagado. El diagrama del circuito es



Solución 3.1 Aplicamos primero el *principio de dualidad*, intercambiando cada variable por su complemento y cada operación de producto por una de suma en las expresiones dadas

$$1.\ (ab' + c)' \qquad 2.\ \left[ad(b' + c)\right]'$$

Obtenemos

$$1.\ (ab' + c)' = (a' + b)c' \qquad 2.\ \left[ad(b' + c)\right]' = a' + d' + bc'.$$

Verificamos desarrollando cada una por medio de las leyes de De Morgan,

$$\begin{aligned}(ab' + c)' &= \left(ab'\right)' c' \\ &= \left[a' + \left(b'\right)'\right] c' \\ &= \left(a' + b\right) c'\end{aligned}$$

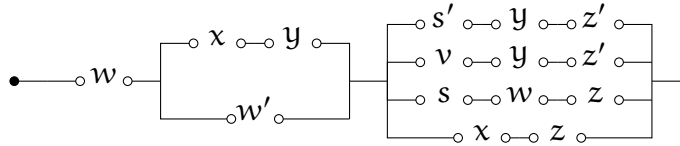
$$\begin{aligned}\left[ad(b' + c)\right]' &= (ad)' + (b' + c)' \\ &= a' + d' + \left(b'\right)' c' \\ &= a' + d' + bc'\end{aligned}$$

Solución 4.1 Tracemos el diagrama correspondiente a la expresión

$$w(xy + w')(s'yz' + vyz' + swz + xz).$$

La expresión tiene tres factores, a saber, w, $(xy + w')$ y $(s'yz' + vyz' + swz + xz)$, que irán colocados en serie. El segundo factor es una suma, luego esa componente es un circuito en paralelo, el primer sumando xy es un producto, que está en serie. así, el segundo factor consta de dos *ramas* en paralelo, una de ellas tiene a las conexiones x y y en serie, y la otra contiene a la conexión w'.

De manera análoga, el tercer factor consta de cuatro sumandos, es decir, cuatro *ramas* en paralelo, cada rama contiene varias conexiones en serie, como lo vemos en la figura:



Ahora simplificaremos la expresión, veamos los dos primeros factores w y $xy + w'$, efectuemos el producto y hagamos operaciones,

$$\begin{aligned} w(xy + w') &= wxy + ww' \\ &= wxy + 0 \\ &= wxy. \end{aligned}$$

Multiplicamos por el tercer factor y obtenemos que la expresión original es igual a

$$wxys'yz' + wxyvyz' + wxyswz + wxyxz.$$

Simplifiquemos cada término,

$$\begin{aligned} wxys'yz' &= wxys'z', \\ wxyvyz' &= wxyvz', \\ wxyswz &= wxysz, \\ wxyxz &= wxyz. \end{aligned}$$

Luego la expresión original es igual a

$$wxys'z' + wxyvz' + wxysz + wxyz.$$

Veamos los dos últimos sumandos:

$$\begin{aligned} wxysz + wxyz &= wxyz + wxyzs \\ &= wxyz + (wxyz)s \\ &= wxyz. \qquad\qquad a + ab = a, \text{ Ejem 3.3, p. 68} \end{aligned}$$

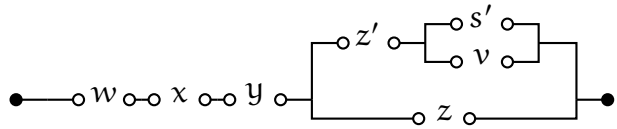
Así, la expresión original queda como

$$\begin{aligned} wxys'z' + wxyvz' + wxyz &= wxy(s'z' + vz' + z) \\ &= wxy\left[z'(s' + v) + z\right]. \end{aligned}$$

Es decir, hemos simplificado la expresión original, obtuvimos

$$wxy\left[z'(s' + v) + z\right].$$

Según nos convenga podemos emplear esta simplificación o tratar se reducirla o transformarla aún más. Nos quedaremos con esta expresión. Ahora obtenemos su diagrama, se trata de cuatro conexiones en serie, las tres primeras son w, x y y, la cuarta es una conexión en paralelo de dos *ramas*, la superior es una conexión en serie, $z'(s' + v)$ donde el segundo producto es una en paralelo.



Solución 4.2 En la expresión $x'z' + xyz + yz' + x'y'z + x'yz$ repetimos el último sumando pues sabemos que $w = w + w$, así, $x'yz = (x'yz) + (x'yz)$, tenemos entonces que

$$x'z' + xyz + yz' + x'y'z + x'yz = x'z' + xyz + yz' + x'y'z + x'yz + x'yz.$$

Reagrupamos el lado derecho anterior como

$$x'z' + yz(x + x') + yz' + x'z(y' + y),$$

sabemos que $x + x' = 1$ y $y' + y = 1$, así, lo anterior es igual a

$$x'z' + yz + yz' + x'z,$$

lo cual es igual a $x'(z' + z) + y(z + z')$ que es igual a $x' + y$.

Solución 4.3 Simplifiquemos $f(x) = (1 - x)x' + (1 + x')x$,

$$\begin{aligned} f(x) &= (1 - x)x' + (1 + x')x \\ &= 1 \cdot x' + x \cdot x' + x + x' \cdot x \\ &= x' + 0 + x + 0 \\ &= x' + x, \end{aligned}$$

lo cual es igual al resultado obtenido.

Solución 4.4 Para demostrar la fórmula por el método de *inducción perfecta*, debemos verificar que al substituir la variable en cuestión por 0 y después por 1, en ambos casos obtenemos una identidad.

Substituimos $x = 0$ en

$$x' + f(x, y, z, ...) = x' + f(1, y, z, ...).$$

Del lado izquierdo obtenemos

$$\begin{aligned} 0' + f(0, y, z, ...) &= 1 + f(0, y, z, ...) \\ &= 1, \end{aligned}$$

y del lado derecho,

$$\begin{aligned} 0' + f(1, y, z, ...) &= 1 + f(1, y, z, ...) \\ &= 1. \end{aligned}$$

Para $x = 0$ se obtiene una identidad.

Substituimos ahora $x = 1$. Del lado izquierdo obtenemos

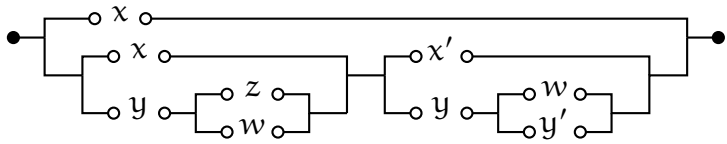
$$1' + f(1, y, z, ...) = 0 + f(1, y, z, ...) = f(1, y, z, ...).$$

Del lado derecho,

$$1' + f(1, y, z, ...) = 0 + f(1, y, z, ...) = f(1, y, z, ...).$$

En ambos casos obtenemos una identidad luego queda demostrada la fórmula.

Solución 4.5 Para simplificar el circuito, primero hallamos la función booleana que lo representa,



La función

$$f(w, x, y, z) = x + \big[x + y(z + w)\big]\,\big[x' + y(w + y')\big]$$

representa al circuito.

La podemos expresar como

$$f(w, x, y, z) = x + g(w, x, y, z)$$

donde $g(w, x, y, z) = \big[x + y(z + w)\big]\,\big[x' + y(w + y')\big]$.

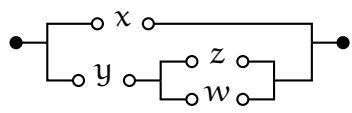
Aplicamos la fórmula (**17b**) de la página 79 y obtenemos

$$\begin{aligned} f(w, x, y, z) &= x + g(w, x, y, z) \\ &= x + g(w, 0, y, z) \\ &= x + \big[0 + y(z + w)\big]\,\big[0' + y(w + y')\big] \\ &= x + \big[y(z + w)\big]\,\big[1 + y(w + y')\big] \\ &= x + \big[y(z + w)\big] \cdot 1 \\ &= x + y(z + w). \end{aligned}$$

Así, el circuito se simplifica a

$$x + \big[x + y(z + w)\big]\,\big[x' + y(w + y')\big] = x + y(z + w)$$

cuyo diagrama es



Solución 4.6 Los resultados obtenidos al simplificar el circuito

$$w(xy + w')(s'yz' + vyz' + swz + xz)$$

fueron, según el procedimiento algebraico empleado en la solución del Problema 4.1,

$$wxy\left[z'(s' + v) + z\right].$$

Y según el ejemplo 4.6, por medio de *Wolfram|Alpha*,

$$wxy(s' + v + z).$$

Pero las expresiones $z'(s' + v) + z$ y $(s' + v + z)$ son equivalentes, veamos

$$\begin{aligned} z'(s' + v) + z &= z + z'(s' + v) \\ &= z + 0'(s' + v) \qquad \text{Por la fórmula } \mathbf{17b} \\ &= z + s' + v \\ &= s' + v + z. \end{aligned}$$

Bibliografía

al-Jwârizmî, Muhammad ibn Musa.
Hidab al-jabr wa'l muqabalah. ca. 825 d. c.
Ed. y trad. por Frederic Rosen.
The Oriental Translation Fund, 1831.
URL: https://bit.ly/2UHuDKU (visitado 18-02-2021).

Barnett, Janet Heine.
Applications of Boolean Algebra: Claude Shannon and Circuit Design.
2009. URL: https://pdfs.semanticscholar.org/b024/36506d72277cdd24cec4c32a97c3f1c3a745.pdf (visitado 18-02-2021).

Billstein, Rick, Shlomo Libeskind y Johnny W. Lott.
MATEMÁTICAS: Un enfoque de resolución de problemas para maestros de educación básica. Trad. por Manuel López Mateos.
México: López Mateos Editores, 2016. ISBN: 978-1523268528.
URL: https://www.amazon.com/dp/1523268522.

Boole, George. *An Investigation of the Laws of Thought*...
An Investigation of the Laws of Thought, on which are founded the Mathematical Theories of Logic and Probabilities.
London: Walton y Maberly, 1854.
URL: https://archive.org/details/bub_gb_KjQCAAAAQAAJ (visitado 18-02-2021).

Buchholz, Werner. «The Word *Byte* Comes of Age...»
En: *Byte Magazine* 2.2 (1977), pág. 144.
URL: https://bit.ly/360i83Q (visitado 18-02-2021).

Chopin, Frédéric. *24 Preludi, Op. 28*. Recital, Yuja Wang. Teatro La Fenice. 3 de abr. de 2017. URL: https://www.youtube.com/watch?v=pSpf9bKK_Zk&feature=youtu.be (visitado 18-02-2021).

Conan Doyle, Sir Arthur. *A Study in Scarlet*. London, New York y Melbourne: Ward, Lock & Co., 1887. URL: http://bit.ly/2GSZ6ku (visitado 18-02-2021).

De Morgan, August. *Formal Logic or The Calculus of Inference, Necessary and Probable*. London: Taylor y Walton, 1847. URL: http://bit.ly/2GUSH9o (visitado 18-02-2021).

Dufort, Antony. *George Boole Monument For Lincoln 2017*. 2017. URL: https://bit.ly/2PvhrYJ (visitado 18-02-2021).

García Lorca, Federico. *Obras Completas*. Vol. I. Aguilar, 1954.

HU, The. *Wolf Totem*. 16 de nov. de 2018. URL: https://www.youtube.com/watch?v=jM8dCGIm6yc (visitado 18-02-2021).

López Mateos, Manuel. *Conjuntos, lógica y funciones*. Tercera edición. México: MLM editor, 2021. URL: https://goo.gl/DZG55y (visitado 18-02-2021).

Peirce, Charles S. *Writings of Charles S. Peirce. A chronological edition*. Ed. por Christian J. W. Kloesel. Vol. 5. 1884–1886. Bloomington e Indianapolis: Indiana University Press, 2020. ISBN: 9780253056672. URL: https://bit.ly/2ZsZdwH (visitado 18-02-2021).

Russell, Bertrand. *Paradoja de Russell — Wikipedia, The Free Encyclopedia*. URL: https://es.wikipedia.org/wiki/Paradoja_de_Russell (visitado 18-02-2021).

Shannon, Claude Elwood. «A Symbolic Analysis of Relay and Switching Circuits». BS. University of Michigan, 1936. URL: https://history-computer.com/Library/Shannon.pdf (visitado 18-02-2021).

Shannon, Claude Elwood.
«A Symbolic Analysis of Relay and Switching Circuits».
En: *Transactions American Institute of Electrical Engineers* 57 (12 dic. de 1938), págs. 713-723. doi: 10.1109/T-AIEE.1938.5057767.
url: http://www.ccapitalia.net/descarga/docs/1938-shannon-analysis-relay-switching-circuits.pdf (visitado 18-02-2021).
— «The Synthesis of Two Terminal Switching Circuits».
En: *Bell System Technical Journal* 28.1 (ene. de 1949), págs. 59-98.
url: https://archive.org/details/bstj28-1-59 (visitado 18-02-2021).
Wolfram|Alpha. computational intelligence. 2019.
url: https://www.wolframalpha.com/.

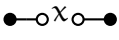
Índice alfabético

Símbolos y notación

bit	Componente electrónico de una computadora que tiene dos estados excluyentes, 0 y 1.	1
byte	es un grupo de 8 bits, usado para codificar información, ya sean caracteres o números.	4
$\in$	es un elemento de, $x \in C$, x es un elemento de C	6
$\notin$	no es un elemento de, $x \notin C$, x no es un elemento de C	6
$\mid$	tal que o tales que	7
Ω	Omega, conjunto universo o total	7
A^c	complemento del conjunto A, $A^c = \{x \in \Omega \mid x \notin A\}$	7
$\overline{A}$	complemento del conjunto A, $\overline{A} = \{x \in \Omega \mid x \notin A\}$	7
A'	complemento del conjunto A, $A' = \{x \in \Omega \mid x \notin A\}$	7
$\subseteq$	subconjunto, $A \subseteq B$, A es subconjunto de B	8
$\emptyset$	conjunto vacío, $\emptyset = \{x \in \Omega \mid x \neq x\}$	9
V ó F	Verdadero o Falso. Posibles ***valores de verdad*** de una proposición.	10
$\neg p$	**no** p. Negación de la proposición p.	10
$\sim$	negación en bits, cambia su estado.	12

	potencial, comunmente llamado *fuente*.	15
NOT	negación en bits, cambia su estado.	17
	compuerta lógica NOT, cambia el valor de entrada por su negativo (lógico).	17
	resistencia, elemento de un circuito eléctrico que impide conexión directa entre dos polos.	18
$A \cap B$	A intersección B es el conjunto $\{ x \in \Omega \mid x \in A \text{ y } x \in B \}$.	18
$p \wedge q$	p ***y*** q es la ***conjunción***, es otra proposición; es verdadera si p es verdadera **y** q es verdadera.	20
AND	operación entre dos en bits, su valor de su estado es 1 sólo si el estado de ambos bits es 1.	21
&	operación entre dos en bits, su valor de su estado es 1 sólo si el estado de ambos bits es 1.	21
	compuerta lógica AND, su valor es 1 sólo si el valor de *ambas* entradas es 1.	23
$\Leftrightarrow$	Si, y sólo si. Equivalencia lógica entre dos proposiciones o afirmaciones. Si la proposición $p \leftrightarrow q$ es una tautología, entonces $p \Leftrightarrow q$.	24
$A \cup B$	A unión B es el conjunto $\{ x \in \Omega \mid x \in A \text{ ó } x \in B \}$.	24
$p \vee q$	p ***ó*** q es la ***disyunción***, es otra proposición; es verdadera si p es verdadera ***o*** q es verdadera, ***o*** ambas lo son.	25
OR	operación entre dos en bits, su valor de su estado es 1 si el estado de algún bit es 1.	26
|	operación entre dos en bits, su valor de su estado es 1 si el estado de algún bit es 1.	26
	compuerta lógica OR, para que su valor sea 1 basta que el valor de *alguna* entrada sea 1.	28
$A \setminus B$	A diferencia B es el conjunto $\{x \in \Omega \mid x \in A \text{ y } x \notin B\}$.	30

$A \triangle B$	A diferencia simétrica B es el conjunto de puntos que están en A o en B pero no en ambos: $(A \setminus B) \cup (B \setminus A)$.	32
$p \veebar q$	***una de dos***, p ***ó*** q es la ***disyunción excluyente***, es otra proposición; es verdadera cuando p es verdadera ***o*** q es verdadera, pero no ambas.	33
`XOR`	operación entre dos en bits, su valor de su estado es `1` si el estado de *sólo* un bit es `1`.	35
`^`	operación entre dos en bits, su valor de su estado es `1` si el estado de *sólo* un bit es `1`.	35
	compuerta lógica `XOR`, para que su valor sea `1` el valor de *sólo una* entrada debe ser `1`.	38
$\equiv$	equivalencia, símbolo de	40
`NAND`	operación entre dos en bits, su valor de su estado es `0` sólo si el estado de ambos bits es `1`.	41
	compuerta lógica `AND`, su valor es `1` sólo si el valor de *ambas* entradas es `1`.	44
`NOR`	operación entre dos en bits, su valor de su estado es `1` si el estado de *ambos* bit es `0`.	46
	compuerta lógica `NOR`, su valor es `1` sólo si el valor de *ambas* entradas es `0`.	50
`XNOR`	operación entre dos en bits, su valor de su estado es `1` si el estado de *ambos* bit es `0` o es `1`.	53
	compuerta lógica `XNOR`, su valor es `1` sólo si el valor de *ambas* entradas es `0` o es `1`.	55
$\mathfrak{B}$	conjunto cuyos elementos cumplen las propiedades de álgebra de Boole.	58
ab	a *por* b, poducto de dos elementos, se usa en varios ámbitos, en álgebra de Boole o en números naturales y otros.	59
$\Rightarrow$	implicación lógica. Si la proposición $p \rightarrow q$ es una tautología, entonces $p \Rightarrow q$.	59
$\forall$	para toda(o), *cuantificador universal*.	59

$\exists$	existe, *cuantificador existencial*.	60
$+$	símbolo de suma, se usa en varios ámbitos, para variables booleanas o para números.	60
$\mathbb{N}$	el conjunto de los números naturales, $\mathbb{N} = \{1, 2, 3, \ldots, n, n+1, \ldots\}$.	62
a'	a *prima*, símbolo de negativo o complemento de una variable booleana.	62
$\oplus$	$a \oplus b = ab' + ba'$, suma excluyente entre variables booleanas, similar a la compuerta XOR.	63
$\preccurlyeq$	precede, relación de orden parcial, $a \preccurlyeq b$, a *precede a* b.	63
$\prec$	precede estrictamente, es decir, precede pero no es igual, relación de orden parcial, similar a *menor que*.	65
	conexión x en un circuito.	72

Manuel López Mateos inició su actividad docente en 1967 en la Facultad de Ciencias de la UNAM. Ha impartido cursos de Cálculo diferencial e integral, Análisis matemático y Álgebra lineal, entre otros. En particular, en el año de 1972, impartió, en el entonces Centro de Didáctica de la UNAM, cursos de capacitación para la primera generación de profesores de matemáticas del Colegio de Ciencias y Humanidades (CCH) de la UNAM. Ha traducido más de 15 importantes libros de texto de matemáticas. En 2003 fue el director fundador de la Facultad de Ciencias de la UABJO.

https://matecompu.mi-libro.club/

MLM
EDITOR
2022