Algebra de Boole, Leyes de DeMorgan

Sitio:	Agencia de Habilidades para el Futuro
Curso:	Lógica Computacional 1°D
Libro:	Algebra de Boole, Leyes de DeMorgan
Imprimido por:	RODRIGO PINTO
Día:	martes, 26 de noviembre de 2024, 10:24

Descripción

Tabla de contenidos

1. Circuitos

Circuitos

El diseño de circuitos electrónicos es el proceso de crear un sistema electrónico funcional que cumpla con ciertos requisitos y especificaciones. Antes de sumergirnos en los detalles, es importante comprender los conceptos básicos del diseño de circuitos.
Un circuito electrónico consiste en una red de componentes interconectados que trabajan juntos para realizar una función específica. Estos componentes incluyen resistencias, condensadores, inductores, transistores y muchos otros. El diseño de un circuito electrónico implica seleccionar los componentes adecuados, conectarlos de manera apropiada y asegurarse de que el circuito funcione de acuerdo con las especificaciones requeridas.

En la materia, no vamos a llegar a trabajar con todos esos componentes, sino que nos vamos a limitar a ver las entradas y, aplicando lo visto hasta el momento, vamos a utilizar las compuertas lógicas conocidas (AND, OR, XOR, NOR, etc.) para diseñar los circuitos con base en un enunciado determinado.

A esta altura, querrán saber:¿para qué sirve el Álgebra de Boole?, y ¿por qué me habla de De Morgan?
Bueno, el álgebra de Boole se utiliza para modelar circuitos electrónicos. Un dispositivo electrónico está constituido por un número determinado de circuitos. Cada circuito puede diseñarse aplicando las reglas del álgebra de Boole. Las Leyes de De Morgan son fundamentales para la construcción y análisis de circuitos lógicos, que son esenciales en el diseño de sistemas digitales como computadoras, controladores y otros dispositivos electrónicos.

2. Relación y Codependencia en la Construcción de Circuitos Lógicos

¿Cuándo una expresión es verdadera?

En el diseño de circuitos lógicos, el álgebra de Boole proporciona la base para describir cómo funcionan las puertas lógicas (AND, OR, NOT, etc.). Las leyes de De Morgan, por otro lado, son fundamentales para la simplificación de expresiones lógicas y para la implementación física de estas en circuitos.

- Simplificación de circuitos: las leyes de De Morgan permiten transformar expresiones lógicas de manera que se minimice el número de puertas lógicas necesarias para implementarlas, lo que es crucial para optimizar el diseño de circuitos.
- Implementación práctica: en algunos casos, ciertas combinaciones de puertas lógicas son más fáciles o eficientes de implementar que otras en hardware. Las leyes de De Morgan permiten reescribir expresiones para adaptarlas a las puertas disponibles. Por ejemplo, si se dispone de puertas NOR en lugar de AND y OR, se puede usar De Morgan para convertir las operaciones en términos de NOR.
- Reducción de costos y mejora de la eficiencia: al simplificar las expresiones lógicas mediante el álgebra de Boole y las leyes de De Morgan, se pueden reducir los costos de fabricación y mejorar la eficiencia energética de los circuitos lógicos.

En resumen, el álgebra de Boole proporciona las reglas básicas para trabajar con variables lógicas, mientras que las leyes de De Morgan ofrecen herramientas para transformar y simplificar esas expresiones, lo cual es crucial en el diseño y optimización de circuitos lógicos.

3. Leyes de De Morgan

Leyes de De Morgan

Las leyes de De Morgan son dos reglas importantes que se utilizan para simplificar expresiones lógicas y son muy útiles en el diseño de circuitos lógicos. Estas leyes son:

Primera ley de De Morgan:

- ¬(A·B) = ¬A + ¬B
- Esta ley establece que la negación de una conjunción (AND) es equivalente a la disyunción (OR) de las negaciones.
- Ejemplo: Supongamos que A= 1 y que B = 1

  A * B = 1

  ¬ (A * B) = 0

Según esta ley, ¬A = 0 y ¬B = 0, por lo tanto, ¬A + ¬B = 0 + 0 = 0

Ahora, "en castellano": el complemento de un producto de variables, es igual a la suma de los complementos de las variables.

Si no quedó claro: El complemento de dos o más variables a las que se aplica la operación AND es equivalente a aplicar la operación OR a los complementos de cada variable.

Segunda ley de De Morgan:

- ¬(A+B)= ¬A * ¬B
- Esta ley establece que la negación de una disyunción (OR) es equivalente a la conjunción (AND) de las negaciones.
- Ejemplo: Suponiendo que A = 0 y B = 1

A + B = 1

¬ (A + B) = 0

Según esta ley, ¬A = 1 y ¬B = 0, entonces ¬A * ¬B = 1 * 0 = 0

En un lenguaje coloquial, esto significa: El complemento de una suma de variables, es igual al producto de los complementos de las variables. O, dicho de otra manera: El complemento de dos o más variables a las que se aplica la operación OR es equivalente a aplicar la operación AND a los complementos de cada variable.

Estas leyes son fundamentales para la simplificación y transformación de expresiones lógicas en circuitos digitales.

Permiten diseñar circuitos más eficientes y minimizan la cantidad de componentes necesarios. Además, son esenciales para entender cómo funcionan las puertas lógicas NOR y NAND, que son muy comunes en la electrónica digital.

4. Álgebra de Boole

Álgebra de Boole

El álgebra de Boole es un sistema matemático que trabaja con variables que pueden tomar dos valores: 0 (falso) y 1 (verdadero). Las operaciones básicas en el álgebra de Boole son:

AND (Conjunción): La operación AND entre dos variables es verdadera (1) solo si ambas son verdaderas.

Ejemplo: $A * B$ o $A \land B$

OR (Disyunción): La operación OR entre dos variables es verdadera si al menos una de ellas es verdadera.

Ejemplo: $A + B$ o $A \lor B$

NOT (Negación): La operación NOT invierte el valor de la variable. Si la variable es verdadera (1), se convierte en falsa (0), y viceversa.

Ejemplo: $\neg A$ ó $\sim A$ (también puede encontrarse con la $\bar{A}$)

Dentro del álgebra de Boole, es de utilidad definir que está compuesta por solo dos elementos. Es un conjunto de elementos binarios relacionados entre sí mediante las operaciones lógicas, producto (*) y suma (+), que cumplen con los siguientes postulados:

ELEMENTO IDENTIDAD

a + 0 = a

a * 1 = a

PROPIEDAD CONMUTATIVA

a + b = b + a

a * b = b * a

PROPIEDAD DISTRIBUTIVA

a * (b + c) = (a * b) + (a * c)

a + (b * c) = (a + b) * (a + c)

INVERSIÓN LÓGICA (ó COMPLEMENTACIÓN)

a + ¬a = 1

a * ¬a = 0

4.1. Teoremas

Teoremas

1) Dualidad: toda igualdad lógica sigue siendo válida si se intercambian los operadores (+ y *) y los elementos de identidad (0 y 1). La simetría de los postulados demuestra este teorema.

2) El álgebra es un conjunto cerrado; es decir, los resultados de aplicar las operaciones lógicas a las variables, pertenecen al álgebra.

3) En el álgebra se cumple que

$a + 1 = 1$
$a * 0 = 0$

4) Ley de Idempotencia

$a + a = a$
$a * a = a$

5) Ley de involución

$(a')' = a$

6) Las operaciones lógicas son asociativas

a + (b + a) = (a + b) + c
a . (b . c) = a . (b . c)

7) Absorción:

a = a + (a . b)
a = a . (a + b)
a + (¬a * b) = (a + b)
a * (¬a + b) = (a * b)

8) Leyes de De Morgan

(a + b + c + d + ...+ n)′ = a′ . b′ . c′ . d′ ....n′
(a . b . c . d ....n)′ = a′ + b′ + c′ + d′ + ...+ n′

9) Asociativa

a- Con respecto a la disyunción

((a + b) + c) = (a + (b + c))

b- Con respecto a la conjunción

((a * b) + c) = ((a + c) * (b + c))

10) Conmutativa

a- Con respecto a la disyunción

(a + b) = (b + a)

b- Con respecto a la conjunción

$(a * b) = (b * a)$

11) Distributiva

a- De la conjunción respecto a la disyunción

$[(a + b) * c] = [(a * c) + (c * b)]$

b- De la disyunción respecto a la conjunción

$[(a * b) + c] = [(a * b) + (b * c)]$

12) Condicional

$a \rightarrow b = \neg a + b$

13) Negación de la Condicional

$\neg(a \rightarrow b) = a * \neg b$

14) Condicional contrarrecíproca

$a \rightarrow b = \neg a \rightarrow \neg b$

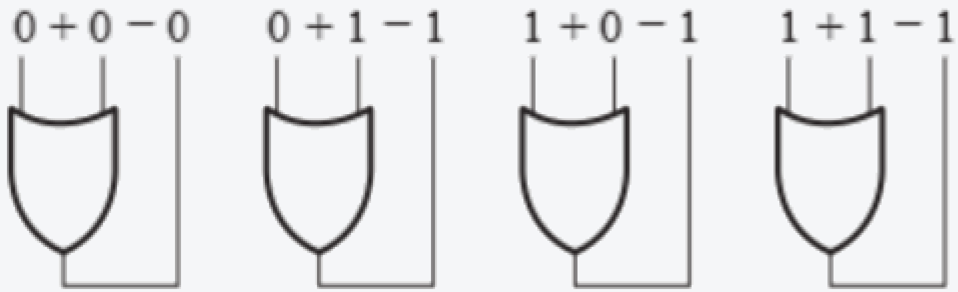
15) Transportación

$(a \rightarrow b) * (b \rightarrow c) = a \rightarrow c$

4.2. Suma Booleana

Suma Booleana

La suma booleana es el equivalente a la operación OR. A continuación, se muestran las reglas básicas junto a con su relación con la compuerta OR:

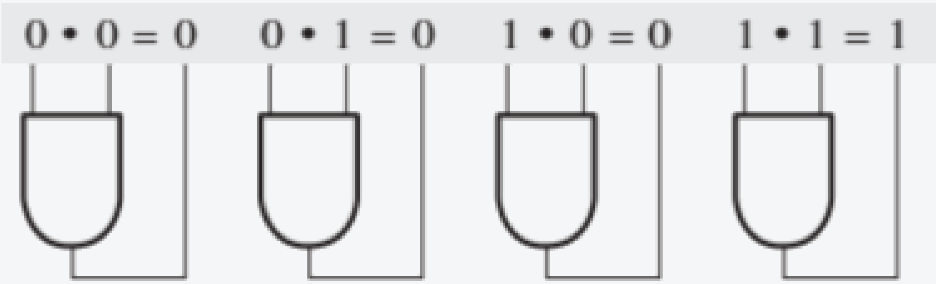


La compuerta OR es el sumador booleano En el álgebra de Boole, un término suma es una suma de literales.
En los circuitos lógicos, un término suma se obtiene mediante una operación OR, sin que exista ninguna operación AND en la expresión.

4.3. Multiplicación booleana

Multiplicación booleana

La multiplicación boolena es equivalente a la operación AND y sus reglas básicas junto con sus relaciones con la puerta AND se ilustran de la siguiente manera:



4.4. Reglas básicas y simplificación lógica

Multiplicación booleana

1. A + 0 = A

Si aplicamos la operación **OR** a una variable cualquiera y a 0, el resultado es siempre igual a la variable. Si A es 1, la salida es igual a 1 y, por tanto, igual a A. Si A es 0, la salida es 0, e igualmente, es idéntica a A.

2. A + 1 = 1

Si se aplica la operación **OR** a una variable y a 1, el resultado es siempre igual a 1. Un 1 en una entrada de una puerta **OR** produce siempre un 1 en la salida, independientemente del valor de la otra entrada.

3. A * 0 = 0

Si se aplica la operación **AND** a una variable y a 0, el resultado es siempre igual a 0. Siempre que una de las entradas de una puerta **AND** sea 0, la salida es siempre 0, independientemente del valor de la otra salida.

4. A * 1 = A

Si aplica la operación **AND** a una variable y a 1, el resultado es siempre igual a la variable. Si la variable A es 0, la salida de la puerta **AND** será siempre 0, mientras que si A es 1, la salida será 1, dado que las dos entradas son 1.

5. A + A = A

Si se aplica la operación **OR** a una variable consigo misma, el resultado es siempre igual a la variable. Si A es 0, entonces 0 + 0 = 0, mientras que si A es 1, 1 + 1 = 1.

6. A + ¬A = 1

Si se aplica la operación **OR** a una variable y a su complemento, el resultado es siempre igual a 1. Si A es 0, entonces 0 + ¬0 = 0 + 1 = 1. Si A es 1, entonces 1 + ¬1 = 1 + 0 = 1.

7. A * A = A

Si se aplica la operación **AND** a una variable consigo misma, el resultado siempre es igual a la variable. Si A = 0, entonces 0 * 0 = 0, y si A = 1, entonces 1 * 1 = 1.

8. A * ¬A = 0

Si se aplica la operación **AND** a una variable y a su complemento, el resultado es siempre igual a 0. Esta regla se basa en que siempre A o ¬A será 0, y además en que cuando se aplica un 0 a una de las entradas de una puerta **AND**, la salida siempre es 0.

9. ¬(¬A) = A

El complemento del complemento de una variable es siempre la propia variable. El complemento de la variable A es ¬A y el complemento de ¬A será de nuevo A, que es la variable original.

10. A + AB = A

A esta regla se la puede obtener aplicando la ley distributiva y las reglas 2 y 4, de la siguiente manera:

A + AB = A(1 + B) *sacar factor común (ley distributiva)*

= A * 1 *regla 2: (1 + B) = 1*

= A *regla 4: A * 1 = A*

11. A + ¬AB = A + B

Esta regla puede demostrarse de la siguiente manera:

A + ¬AB = (A + AB) + ¬AB *regla 10: A = A + AB*

= (AA + AB) + ¬AB *regla 7: A = AA*

= AA + AB + A¬A + ¬AB *regla 8: sumar A¬A=0*

= (A + ¬A) (A + B) *sacar factor común*

= 1 * (A + B) *regla 6: A + ¬A = 1*

= A + B *regla 4: eliminar el 1*

12. (A + B) * (A + C) = A + BC

Esta regla puede demostrarse de la siguiente manera:

(A + B) * (A + C) = AA + AC + AB + BC *Ley distributiva*

= A + AC + AB + BC *regla 7: AA = A*

= A (1 + C) + AB + BC *sacar factor común (Ley distributiva)*

= A * 1 + AB + BC *regla 2: 1 + C = 1*

= A (1 + B) + BC *sacar factor común (Ley distributiva)*

= A * 1 + BC *regla 2: 1 + B = 1*

= A + BC *regla 4: A * 1 = A*