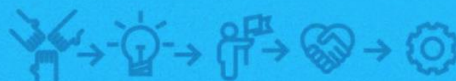




#SOMOS
MAIS
UNIFG





George Boole

- nasceu em Lincoln - Inglaterra em 2 de Novembro de 1815, filho de um sapateiro pobre.
- formação base na escola primária da National Society foi muito rudimentar.
- autodidata, fundou aos 20 anos de idade a sua própria escola e dedicou-se ao estudo da Matemática.



- Em 1840 publicou o seu primeiro trabalho original e em 1844 foi condecorado com a medalha de ouro da Royal Society pelo seu trabalho sobre cálculo de operadores.
- Em 1847 publica um volume sob o título The Mathematical Analysis of Logic em que introduz os conceitos de lógica simbólica demonstrando que a lógica podia ser representada por equações algébricas.



- Este trabalho é fundamental para a construção e programação dos computadores eletrônicos iniciada cerca de 100 anos mais tarde.

- **Aplicação**

. Atualmente todos os computadores usam a Álgebra de Boole materializada em microchips que contêm milhares de interruptores miniaturizados combinados em portas (gates) lógicos que produzem os resultados das operações utilizando uma linguagem binária.

Sinais binários

- . A lógica digital esconde a realidade analógica ao mapear uma gama infinita de valores reais em apenas 2 valores: 0 e 1.
- . A um valor lógico, 0 ou 1, é comum chamar-se um dígito binário (bit).
- . Com n bits, pode representar-se 2^n entidades distintas.
- . quando um projetista lida com circuitos electrónicos, é comum usar os termos “BAIXO” e “ALTO”, em vez de “0” e “1”.



- . Considerar que 0 é BAIXO e 1 é ALTO, corresponde a usar lógica positiva.
- . Considerar que 0 é ALTO e 1 é BAIXO é designada de lógica negativa.

Sistemas Combinacionais X Sequenciais

- . Um sistema lógico combinacional é aquele em que as saídas dependem apenas do valor atual das entradas.
- . pode ser descrito por uma tabela de verdade.



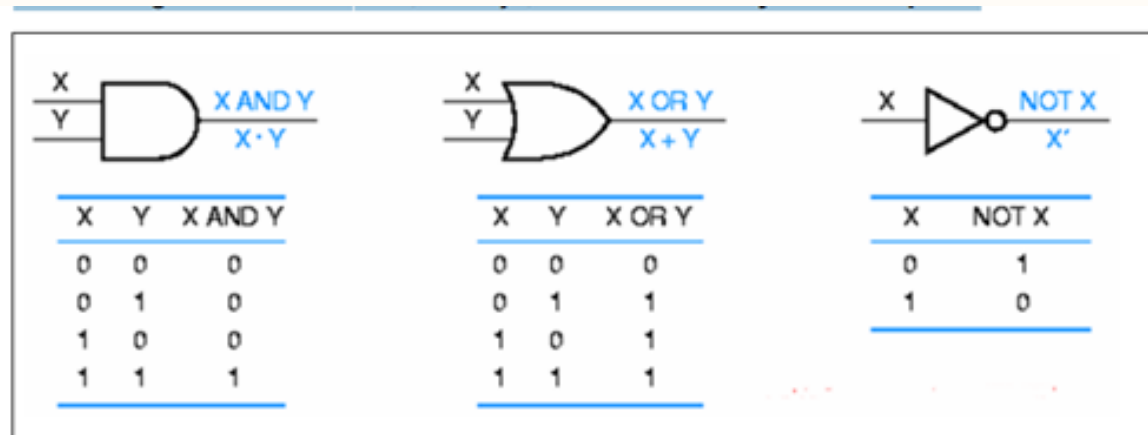
- . Além do valor atual das entradas, as saídas de um circuito lógico sequencial dependem também da sequência de valores por que passaram as entradas \Rightarrow memória.
- . Um sistema sequencial pode ser descrito através duma tabela de estados.
- . Um sistema combinacional pode conter qualquer número de portas lógicas mas não ciclos de realimentação (feedback loops).



- . Um ciclo de realimentação é um caminho do circuito, que permite a um sinal de saída de uma porta ser propagado de volta, para a entrada dessa porta.
- . Regra geral, os ciclos de realimentação introduzem um comportamento sequencial nos circuitos.

Portas lógicas Com 3 tipos de porta elementares (AND, OR, NOT)

. consegue construir-se qualquer sistema digital combinacional, formam um conjunto completo.



Os símbolos e as tabelas de verdade do AND e do OR podem ser generalizados para portas com qualquer número de entradas. A bolha na saída do inversor representa um comportamento "invertido". Combinando numa única porta, um NOT com uma função AND ou OR, obtêm-se 2 novas funções lógicas: NAND e NOR.



X	Y	X NAND Y
0	0	1
0	1	1
1	0	1
1	1	0



X	Y	X NOR Y
0	0	1
0	1	0
1	0	0
1	1	0

FUNCIONALIDADE - Álgebra Booleana


. Para descrever os circuitos que podem ser construídos pela combinação de portas lógicas, um novo tipo de álgebra é necessário, uma em que as variáveis e funções podem ter apenas valores 0 e 1.

. Álgebra booleana possui uma ou mais variáveis de entrada e fornece somente um resultado que depende apenas dos valores destas variáveis.



- . uma função de n variáveis possui apenas 2^n conjuntos possíveis de valores de entrada,
- . a função pode ser descrita completamente através de uma tabela de 2^n linhas,
- . cada linha mostrando o valor da função para uma combinação diferente dos valores de entrada.
- . Tabela é denominada tabela verdade.

A B C 0 0 0 0 1 0 1 0 0 1 1 1



. tabela verdade de uma função básica a função
AND

- conjunto de funções da álgebra booleana
- têm implementação eletrônica através de transistores
- são conhecidas como portas lógicas

. um circuito digital é regido pela álgebra de Boole

. com as portas lógicas existentes é possível implementar qualquer função da álgebra booleana.

As principais portas lógicas, simbologia e tabela verdade.

-NOT

A função NOT é implementada na conhecida porta inversora.

A B 0 1 1 0 (a)

(b)

(a) tabela verdade, (b) símbolo



-AND

A função AND pode ser definida em linguagem natural como 1 se todas as entradas forem 1 e 0 se apenas uma das entradas for 0.

A B S 0 0 0 0 1 0 1 0 0 1 1 1



A função OR também pode ser definida em linguagem natural ela é 0 se todas as entradas forem 0 e 1 se existir uma entrada em 1.

A B C 0 0 0 0 1 1 1 0 1 1 1 1

-XOR

A função XOR conhecida como exclusive OR é muito parecido com a OR.

A B C 0 0 0 0 1 1 1 0 1 1 1 1

Considerações parciais ...



- . temos acima algumas das principais portas lógicas existentes
- . não são as únicas mas as outras portas existentes são combinações destas portas básicas
- . todos os circuitos digitais podem ser montados somente com estas portas

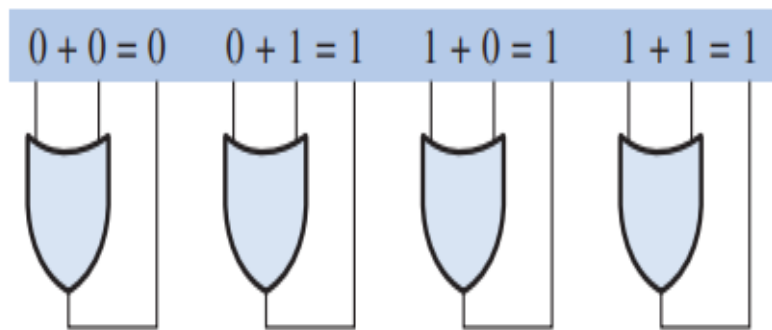
Aprofundando os Fundamentos ...

- . Os termos **variável**, **complemento** e **literal** são usados em álgebra Booleana.
 - **variável**: é um símbolo (geralmente uma letra maiúscula em itálico) usado para representar uma grandeza lógica.
 - . qualquer variável simples pode ter um valor 1 ou 0.
 - **complemento**: é o inverso de uma variável e é indicado por uma barra sobre a variável.

Ex. o complemento da variável A é \bar{A} –



- . O complemento de uma variável A é lido como “ A negado” ou “ A barrado”.
- . Ou ... outro símbolo, em vez de uma barra, para indicar o complemento de uma variável; por exemplo, B' indica o complemento de B .
- **literal:** é a variável ou o complemento de uma variável.
- **Adição Booleana** - equivalente à operação OR e as regras básicas são ilustradas com suas relações com a porta OR da seguinte forma:



Na álgebra Booleana, um **termo-soma** é uma soma de literais. Em circuitos lógicos, um termo-soma é produzido por uma operação OR sem o envolvimento de operações AND. Alguns exemplos de termos-soma são $A + B$, $A + \bar{B}$, $A + B + \bar{C}$ e $\bar{A} + B + C + \bar{D}$.

Um termo-soma será igual a 1 quando uma ou mais das literais no termo for 1. Um termo-soma será igual a 0 somente se cada uma das literais for 0.

Determine os valores de A , B , C e D que tornam o termo-soma $A + \bar{B} + C + \bar{D}$ igual a 0.

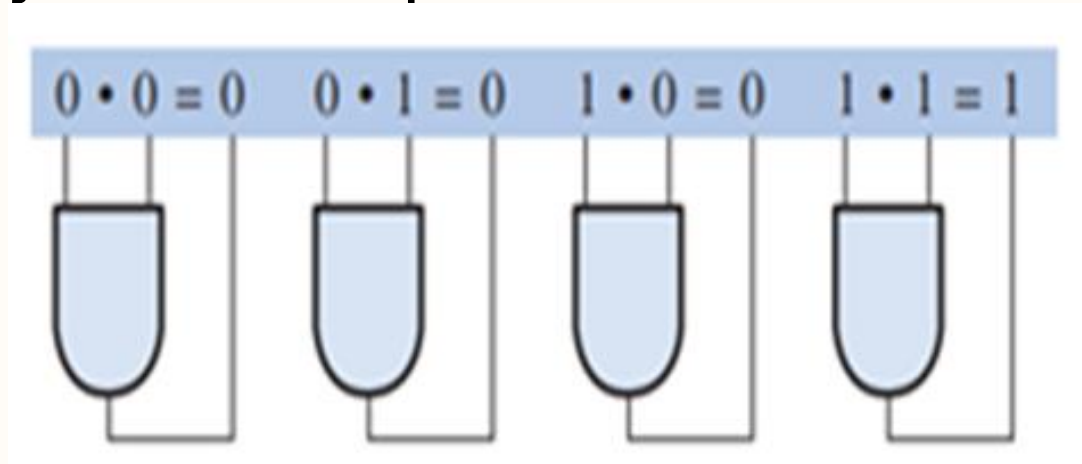
Solução Para o termo-soma ser 0, cada uma das literais tem que ser 0. Portanto, $A = 0$ e $B = 1$, de forma que, $\bar{B} = 0$, $C = 0$ e $D = 1$, de forma que $\bar{D} = 0$.

$$A + \bar{B} + C + \bar{D} = 0 + \bar{1} + 0 + \bar{1} = 0 + 0 + 0 + 0 = 0$$

Problema relacionado* Determine os valores de \bar{A} e B que tornam o termo-soma igual a 0.

Multiplicação Booleana

- . Equivalente à operação AND
- . A porta AND é um multiplicador booleano
- . são relações com a porta AND



Na álgebra Booleana, um **termo-produto** é o produto de literais. Em circuitos lógicos, um termo-produto é produzido por uma operação AND sem o envolvimento de operações OR. Alguns exemplos de termos-produto são AB , $A\bar{B}$, ABC e $\bar{A}\bar{B}\bar{C}\bar{D}$.

Um termo-produto é igual a 1 apenas se cada uma das literais no termo for 1. Um termo-produto é igual a 0 quando uma ou mais das literais for 0.

Determine os valores de A , B , C e D que torna o termo-produto $\bar{A}\bar{B}\bar{C}\bar{D}$ igual a 1.

Solução Para o termo-produto ser 1, cada uma das literais no termo tem que ser 1. Portanto, $A = 1$, $B = 0$ de forma que $\bar{B} = 1$, $C = 1$ e $D = 0$ de forma que $\bar{D} = 1$.

$$\bar{A}\bar{B}\bar{C}\bar{D} = 1 \cdot \bar{0} \cdot 1 \cdot \bar{0} = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

Problema relacionado Determine os valores de A e B que tornam o termo-produto $\bar{A}\bar{B}$ igual a 1.



Leis da Álgebra Booleana

As leis básicas da álgebra Booleana:

- . leis comutativas: para a adição e multiplicação
- . leis associativas: para a adição e multiplicação
- . lei distributiva: são as mesmas que para a álgebra comum.

Cada uma das leis está ilustrada com duas ou três variáveis, porém o número de variáveis não é limitado para essas leis.

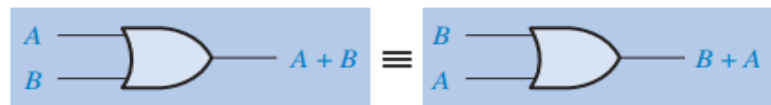
Lei Comutativa A lei comutativa da adição para duas variáveis é escrita da seguinte forma:

$$A + B = B + A$$

Essa lei diz que a ordem das variáveis na qual a função OR é aplicada não faz diferença. Lembre-se que, na álgebra Booleana aplicada a circuitos lógicos, a adição e a operação OR são as mesmas. A Figura 4–1 ilustra a lei comutativa aplicada a uma porta OR e mostra que não importa em qual entrada cada variável é aplicada. (O símbolo \equiv significa “equivalente a”).

► **FIGURA 4–1**

Aplicação da lei comutativa da adição.



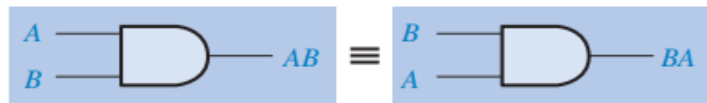
A lei comutativa da multiplicação para duas variáveis é a seguinte:

$$AB = BA$$

Essa lei diz que a ordem das variáveis na qual a operação AND é aplicada não faz diferença. A Figura 4–2 ilustra essa lei aplicada a uma porta AND.

► **FIGURA 4–2**

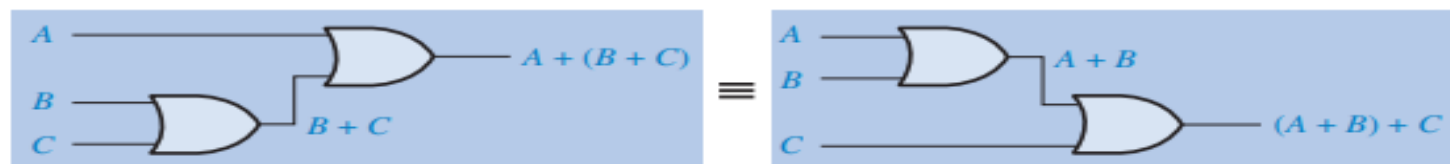
Aplicação da lei comutativa da multiplicação.



Lei Associativa A lei associativa da adição escrita para três variáveis é mostrada a seguir:

$$A + (B + C) = (A + B) + C$$

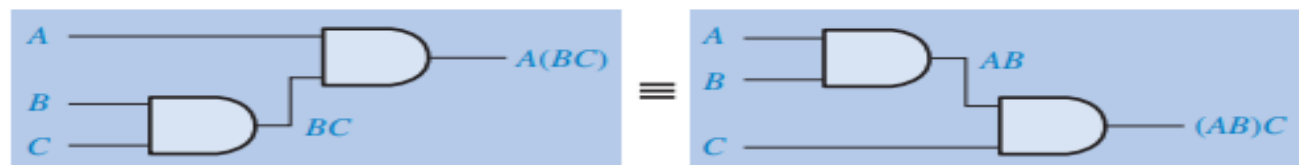
Essa lei diz que quando é aplicada uma operação OR em mais de duas variáveis, o resultado é o mesmo independente da forma de agrupar as variáveis. A Figura 4–3 ilustra essa lei aplicada em portas OR de 2 entradas.



A lei associativa da multiplicação escrita para três variáveis é mostrada a seguir:

$$A(BC) = (AB)C$$

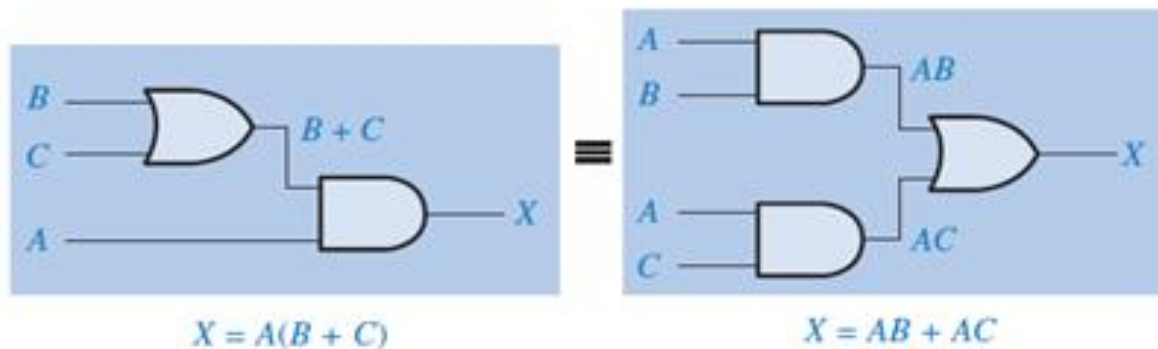
Essa lei diz que a ordem em que as variáveis são agrupadas não faz diferença quando é aplicada uma operação AND em mais de duas variáveis. A Figura 4–4 ilustra essa lei aplicada a portas AND de 2 entradas.



Lei Distributiva A lei distributiva escrita para três variáveis é mostrada a seguir:

$$A(B + C) = AB + AC$$

Essa lei diz que a operação AND de uma única variável com o resultado de uma operação OR aplicada em duas ou mais variáveis é equivalente a uma operação OR entre os resultados das operações AND entre uma única variável e cada uma das duas ou mais variáveis. A lei distributiva também expressa o processo de *fatoração* no qual a variável comum A é fatorada em termos-produto, por exemplo, $AB + AC = A(B + C)$. A Figura 4-5 ilustra a lei distributiva em termos de implementação com portas.



Regras da Álgebra Booleana



A tabela a seguir apresenta uma lista de 12 regras básicas úteis na manipulação e simplificação de expressões Booleanas.

As regras de 1 a 9 serão analisadas em termos de suas aplicações em portas lógicas.

$$1. A + 0 = A$$

$$2. A + 1 = 1$$

$$3. A \cdot 0 = 0$$

$$4. A \cdot 1 = A$$

$$5. A + A = A$$

$$6. A + \bar{A} = 1$$

$$7. A \cdot A = A$$

$$8. A \cdot \bar{A} = 0$$

$$9. \bar{\bar{A}} = A$$

As regras de 10 a 12 serão obtidas em termos de regras mais simples e das leis discutidas anteriormente.

$$10. A + AB = A$$

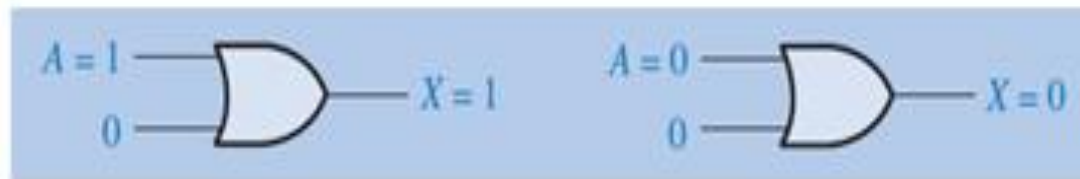
$$11. A + \overline{A}B = A + B$$

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

Propriedade	Complemento	Adição	Multiplicação
Identidade	$\overline{\bar{A}} = A$	$A + 0 = A$ $A + 1 = 1$ $A + A = A$ $A + \bar{A} = 1$	$A \cdot 0 = 0$ $A \cdot 1 = A$ $A \cdot A = A$ $A \cdot \bar{A} = 0$
Comutativa		$A + B = B + A$	$A \cdot B = B \cdot A$
Associativa		$A + (B + C) = (A + B) + C$ $= A + B + C$	$A \cdot (B \cdot C) = (A \cdot B) \cdot C =$ $A \cdot B \cdot C$
Distributiva		$A + (B \cdot C)$ $=$ $(A + B) \cdot (A + C)$	$A \cdot (B + C)$ $=$ $A \cdot B + A \cdot C$

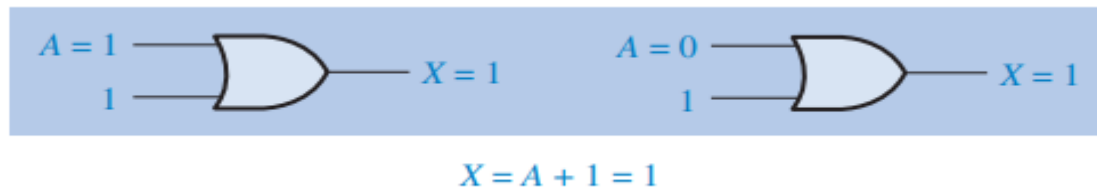
REGRAS DA ÁLGEBRA DE BOOLE

Regra 1. $A + 0 = A$ A operação OR de uma variável com 0 é sempre igual a variável. Se a variável de entrada A for 1, a variável X de saída será 1, que é igual a A . Se A for 0, a saída será 0, que também é igual a A . Essa regra é ilustrada na Figura 4-6, na qual a entrada inferior da porta está fixa em 0.

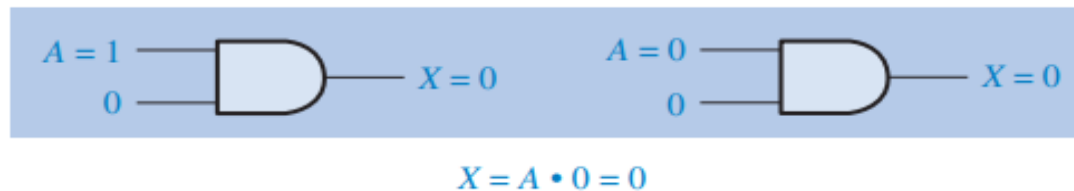


$$X = A + 0 = A$$

Regra 2. $A + 1 = 1$ A operação OR da variável com 1 é igual a 1. Um 1 numa entrada de uma porta OR produz um 1 na saída, independente do valor da variável na outra entrada. Essa regra é ilustrada na Figura 4-7, na qual a entrada inferior da porta está fixa em 1.



Regra 3. $A \cdot 0 = 0$ A operação AND da variável com 0 sempre é igual a 0. Todas as vezes que uma entrada de uma porta AND for 0, a saída será 0, independente do valor da variável na outra entrada. Essa regra está ilustrada na Figura 4-8, na qual a entrada inferior está fixa em 0.

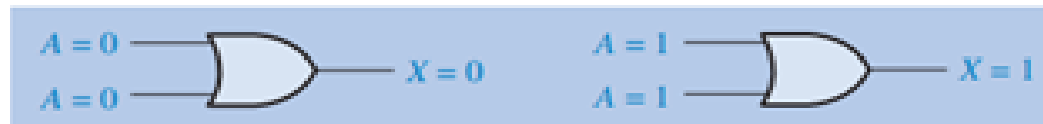


Regra 4. $A \cdot 1 = A$ A operação AND da variável com 1 é sempre igual a variável. Se A for 0 a saída da porta AND será 0. Se A for 1, a saída da porta AND será 1 porque ambas as entradas agora são 1s. Essa regra é mostrada na Figura 4–9, onde a entrada inferior está fixa em 1.



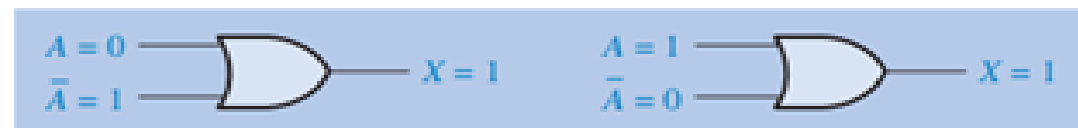
$$X = A \cdot 1 = A$$

Regra 5. $A + A = A$ A operação OR da variável com ela mesma é sempre igual a variável. Se A for 0, então $0 + 0 = 0$; e se A for 1, então $1 + 1 = 1$. Isso é mostrado na Figura 4–10, onde as duas entradas são a mesma variável.



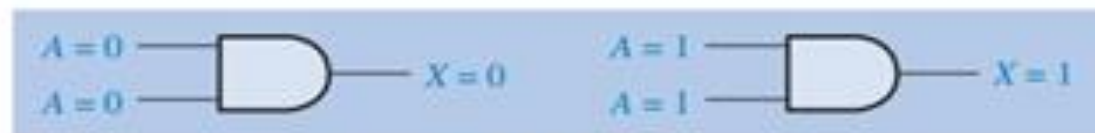
$$X = A + A = A$$

Regra 6. $A + \bar{A} = 1$ A operação OR da variável com o seu complemento é sempre igual a 1. Se A for 0, então $0 + \bar{0} = 0 + 1 = 1$. Se A for 1, então $1 + \bar{1} = 1 + 0 = 1$. Veja a Figura 4-11, onde uma entrada é o complemento da outra.



$$X = A + \bar{A} = 1$$

Regra 7. $A \cdot A = A$ A operação AND de uma variável com ela mesma é sempre igual a variável. Se $A = 0$, então $0 \cdot 0 = 0$; e se $A = 1$, então $1 \cdot 1 = 1$. A Figura 4-12 ilustra essa regra.

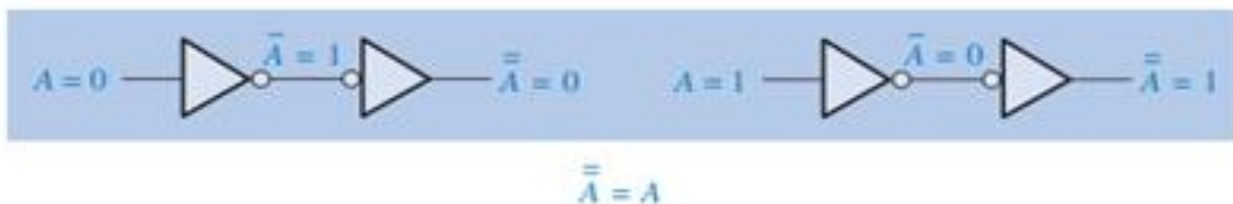


$$X = A \cdot A = A$$

Regra 8. $A \cdot \bar{A} = 0$ A operação AND de uma variável e o seu complemento é sempre igual a 0. Nesse caso, ou \bar{A} ou sempre será 0; e quando um 0 é aplicado na entrada de uma porta AND, a saída também será 0. A Figura 4-13 ilustra essa regra.



Regra 9. $\bar{\bar{A}} = A$ O complemento duplo de uma variável é sempre igual a variável. Se complementarmos (invertamos) a variável A uma vez, obtemos \bar{A} . Então se complementarmos (invertamos) \bar{A} , obtemos A , que é a variável original. Essa regra é mostrada na Figura 4-14 usando inversores.



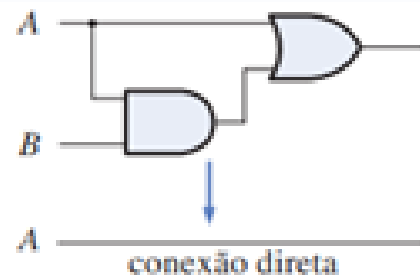
Regra 10. $A + AB = A$ Essa regra pode ser provada aplicando a lei distributiva, Regra 2, e a Regra 4 como a seguir:

$$\begin{aligned} A + B &= A(1 + B) && \text{Fatorando (lei distributiva)} \\ &= A \cdot 1 && \text{Regra 2: } (1 + B) = 1 \\ &= A && \text{Regra 4: } A \cdot 1 = A \end{aligned}$$

A prova é mostrada na Tabela 4–2, onde temos a tabela-verdade e a consequente simplificação do circuito lógico.

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

↑ igual ↑



Regra 11. $A + \bar{A}B = A + B$ Essa regra pode ser provada da seguinte forma:

$$\begin{aligned}
 A + \bar{A}B &= (A + AB) + \bar{A}B \\
 &= (AA + AB) + \bar{A}B \\
 &= AA + AB + A\bar{A} + \bar{A}B \\
 &= (A + \bar{A})(A + B) \\
 &= 1 \cdot (A + B) \\
 &= A + B
 \end{aligned}$$

Regra 10: $A = A + AB$

Regra 7: $A = AA$

Regra 8: adicionando $A\bar{A} = 0$

Fatorando

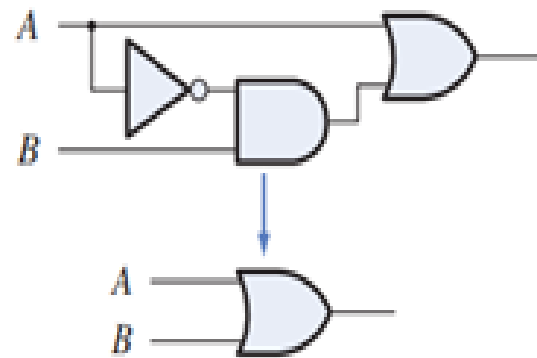
Regra 6: $A + \bar{A} = 1$

Regra 4: simplifica o 1

A prova é mostrada na Tabela 4-3, onde temos a tabela-verdade e a consequente simplificação do circuito lógico.

A	B	$\overline{A}B$	$A + \overline{A}B$	$A + B$
0	0	0	0	0
0	1	1	1	1
1	0	0	1	1
1	1	0	1	1

↑ igual ↑



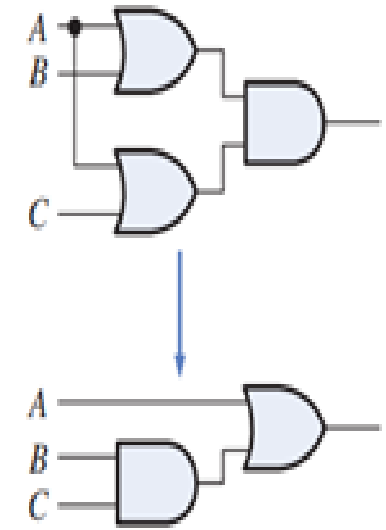
Regra 12. $(A + B)(A + C) = A + BC$ Essa regra pode ser provada da seguinte forma:

$(A + B)(A + C) = AA + AC + AB + BC$	Lei distributiva
$= A + AC + AB + BC$	Regra 7: $AA = A$
$= A(1 + C) + AB + BC$	Fatorando (lei distributiva)
$= A \cdot 1 + AB + BC$	Regra 2: $1 + C = 1$
$= A(1 + B) + BC$	Fatorando (lei distributiva)
$= A \cdot 1 + BC$	Regra 2: $1 + B = 1$
$= A + BC$	Regra 4: $A \cdot 1 = A$

A prova é mostrada na Tabela 4-4, onde temos a tabela-verdade e a conseqüente simplificação do circuito lógico.

A	B	C	$A + B$	$A + C$	$(A + B)(A + C)$	BC	$A + BC$
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1


↑ igual ↑



RESUMO

- . Na Álgebra de Boole existem apenas três operadores E, OU e NÃO (AND, OR, NOT).
- . Estas três funções são as únicas operações necessárias para efetuar comparações ou as quatro operações aritméticas base.





. Em 1937: 75 anos após a morte de Boole, Claude Shannon, estudante no MIT - Boston, USA - estabeleceu a relação entre a Álgebra de Boole e os circuitos eletrônicos transferindo os dois estados lógicos (SIM e NÃO) para várias diferenças de potencial no circuito.

. Todos os computadores usam a Álgebra de Boole materializada em microchips que contêm milhares de interruptores miniaturizados combinados em portas (gates) lógicas que produzem os resultados das operações utilizando uma linguagem binária.



#SOMOS
MAIS
UNIFG