

Circuitos digitais

André Furlan - andre.furlan@unesp.br

Universidade Estadual Paulista Júlio de Mesquita Filho

2024

Sistema binário, álgebra de Boole e lógica

Vantagens

- ▶ Dois estados (mais difícil deteriorar)
- ▶ Discretos
- ▶ Manipulação algébrica (álgebra de Boole)

Mundo ideal: Junção do analógico com o digital

Desvantagens

- ▶ Dois estados em oposição a infinitos estados
- ▶ Discretos em oposição a contínuo

Sistema numérico posicional

Dada a base com uma certa quantidade de elementos B e símbolos

$d \in D = \{d_0, \dots, d_n\}$, então um valor numérico decimal V **posicional** pode ser representado por

$$V(B) = \sum_{i=0}^n d_i \cdot B^{n-i} \quad (1)$$

Sendo $||$ o operador de cardinalidade então $n = |D| - 1$

Por exemplo, para o número **decimal** 249:

$$B = 10$$

$$D = \{2, 4, 9\} \implies |D| = 3 \implies n = 2$$

$$V(10) = \sum_{i=0}^2 d_i \cdot 10^{2-i} =$$
$$d_0 \cdot 10^{2-0} + d_1 \cdot 10^{2-1} + d_2 \cdot 10^{2-2} = \quad (2)$$

$$2 \cdot 10^2 + 4 \cdot 10^1 + 9 \cdot 10^0 =$$

$$\boxed{2 \cdot 100 + 4 \cdot 10 + 9 \cdot 1 = 249}$$

Sistema numérico posicional

Dada a base com uma certa quantidade de elementos B e símbolos

$d \in D = \{d_0, \dots, d_n\}$, então um valor numérico decimal V **posicional** pode ser representado por

$$V(B) = \sum_{i=0}^n d_i \cdot B^{n-i} \quad (1)$$

Sendo $||$ o operador de cardinalidade então $n = |D| - 1$

Por exemplo, para o número **binário** 11:

$$B = 2$$

$$D = \{1, 1\} \implies |D| = 2 \implies n = 1$$

$$\begin{aligned} V(2) &= \sum_{i=0}^1 d_i \cdot 2^{1-i} = \\ d_0 \cdot 2^{1-0} + d_1 \cdot 2^{1-1} &= \\ 1 \cdot 2^1 + 1 \cdot 2^0 &= \\ \boxed{2 + 1 = 3} \end{aligned} \quad (2)$$

Antes de avançarmos no estudo e na representação de circuitos lógicos, vamos entender como os estados de "ligado" e "desligado" dos componentes podem ser utilizados para gerar resultados úteis. Portanto, antes de abordarmos esses tópicos, faremos uma imersão na álgebra booleana, explorando tabelas verdade e formas de composição.

Álgebra de Boole: Operadores **AND**, **OR** e **NOT**

Sendo o sistema binário posicional aplica-se a ele muitas das regras as quais podemos aplicar aos números decimais o que nos leva a Álgebra de Boole e seus operadores:

AND denotado como $.$ ou \wedge

O operador **AND** é uma função de 2 variáveis definida como:

$$AND(a, b) = a.b$$

$$AND(a, b) = 1, \forall a = b = 1 \quad (3)$$

$$AND(a, b) = 0, \forall a \neq b$$

a	b	a . b
0	0	0
0	1	0
1	0	0
1	1	1

Tabela: Tabela verdade para **AND**.

Álgebra de Boole: Operadores **AND**, **OR** e **NOT**

Sendo o sistema binário posicional aplica-se a ele muitas das regras as quais podemos aplicar aos números decimais o que nos leva a Álgebra de Boole e seus operadores:

OR denotado com $+$ ou \vee

O operador **OR** é uma função de 2 variáveis definida como:

$$OR(a, b) = a + b$$

$$OR(a, b) = 0, \forall a = b = 0 \quad (3)$$

$$OR(a, b) = 1, \forall a \neq b, a = b = 1$$

a	b	a + b
0	0	0
0	1	1
1	0	1
1	1	1

Tabela: Tabela verdade para **OR**.

Álgebra de Boole: Operadores **AND**, **OR** e **NOT**

Sendo o sistema binário posicional aplica-se a ele muitas das regras as quais podemos aplicar aos números decimais o que nos leva a Álgebra de Boole e seus operadores:

NOT denotado com \bar{a} ou $\neg a$

O operador **NOT** é uma função de 1 variável definida como:

$$\begin{aligned} NOT(a) &= \bar{a} \\ NOT(a) &= 0, \forall a = 1 \\ NOT(a) &= 1, \forall a = 0 \end{aligned} \quad (3)$$

a	\bar{a}
0	1
1	0

Tabela: Tabela verdade para **NOT**.

Álgebra de Boole: Composição de operadores

É importante destacar que assim com na álgebra tradicional os operadores tem prioridade no momento de sua aplicação na seguinte ordem: **NOT**, **AND** e por fim **OR**. Se usarmos os sinais as regras ficam bem parecidas com a álgebra tradicional: \bar{a} , $.$ e $+$.

A partir dos operadores apresentados vamos brincar um pouquinho com eles:

$$(a + b).c \quad (4)$$

a	b	c	$a+b$	$a + b . c$
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	1	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	0
1	1	1	1	1

Tabela: Tabela verdade para $(a + b) . c$.

Álgebra de Boole: Composição de operadores NAND e NOR

Alguns operadores comuns são compostos:

O operador **NAND** (NOT AND) é definido segundo a seguinte tabela verdade:

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

Tabela: Tabela verdade para **NAND**.

O operador **NOR** (NOT OR) é definido segundo a seguinte tabela verdade:

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

Tabela: Tabela verdade para **NOR**.

Álgebra de Boole: Expressão booleanas equivalentes

Uma expressão booleana equivalente é aquela que, dada uma mesma entrada, retorna exatamente o mesmo resultado. Expressões booleanas equivalentes podem ser ou não a versão otimizada uma da outra.

O que foi dito acima só é possível se alguns **axiomas**¹ forem definidos. A partir desses axiomas podemos definir **teoremas**².

Veremos ambos no próximo slide.

¹hipóteses básicas ou pré-supostos baseados na experiência empírica ou filosófica

²Teoremas são proposições demonstráveis a partir dos axiomas

Álgebra de Boole: Expressão booleanas equivalentes

Axiomas e teoremas

Axiomas:

1. $0.0 = 0$
2. $1.1 = 1$
3. $0.1 = 1.0 = 0$
4. $1 + 1 = 1$
5. $0 + 0 = 0$
6. $1 + 0 = 0 + 1 = 1$
7. $x = 0 \implies \bar{x} = 1$
8. $x = 1 \implies \bar{x} = 0$

Teoremas para AND

1. $x.0 = 0$
2. $x.1 = x$
3. $x.x = x$
4. $x.\bar{x} = 0$

Teoremas para OR

1. $x + 1 = 1$
2. $x + 0 = x$
3. $x + x = x$
4. $x + \bar{x} = 1$

Teorema para NOT

1. $\overline{(\bar{x})} = x$

Álgebra de Boole: Expressão booleanas equivalentes

Princípio da dualidade

Dada uma expressão lógica que expressa uma igualdade então é possível criar uma **expressão dual** na qual a igualdade continua verdadeira.

Uma expressão dual é obtida quando se troca os valores de 0 para 1, de 1 para 0 e os operadores de and para or e de or para and. Perceba que **o resultado da expressão pode mudar** porém a **igualdade é preservada**.

Voltando ao slide 10 você perceberá que os teoremas para *OR* são **dualidades de uma variável** dos teoremas de *AND* e vice-versa.

Álgebra de Boole: Expressão booleanas equivalentes

Dualidade de duas variáveis

$$x.y = y.x \Leftrightarrow x + y = y + x \text{ comutacao}$$

$$x + (x.y) = x \Leftrightarrow x.(x + y) = x \text{ absorcao}$$

$$(x.y) + (x.\bar{y}) = x \Leftrightarrow (x + y).(x + \bar{y}) = x \text{ combinacao} \quad (5)$$

$$\text{Teorema de De Morgan: } \overline{(x.y)} = \bar{x} + \bar{y} \Leftrightarrow \overline{(x + y)} = \bar{x}.\bar{y}$$

$$x + (\bar{x}.y) = x + y \Leftrightarrow x.(\bar{x} + y) = x.y$$

Álgebra de Boole: Expressão booleanas equivalentes

Dualidade de três variáveis

$$x.(y.z) = (x.y).z \Leftrightarrow x + (y + z) = (x + y) + z \text{ associação}$$

$$x.(y + z) = (x.y) + (x.z) \Leftrightarrow x + (y.z) = (x + y).(x + z) \text{ distribuição}$$

$$(x.y) + (y.z) + (\bar{x}.z) = (x.y) + (\bar{x}.z) \Leftrightarrow (x + y).(y + z).(\bar{x} + z) = (x + y).(\bar{x} + z)$$

consenso (sumiço :D)

(6)

Em tempo: Eu sei que em alguns lugares tem parenteses sobrando, isso foi intencional pra facilitar a visualização das relações.

Teorema de D.Morgan

Tabela: Prova exaustiva do teorema de De Morgan: $\overline{(x \cdot y)} = \bar{x} + \bar{y}$

x	y	$x \cdot y$	$\overline{x \cdot y}$	\bar{x}	\bar{y}	$\bar{x} + \bar{y}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

Tabela: Prova exaustiva para o dual do teorema de De Morgan: $\overline{(x + y)} = \bar{x} \cdot \bar{y}$

x	y	$x + y$	$\overline{x + y}$	\bar{x}	\bar{y}	$\bar{x} \cdot \bar{y}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Provas de expressões na álgebra de Boole

- ▶ indução perfeita: Devido a pequena quantidade de valores possíveis (0 ou 1) geralmente é viável provar as igualdades dessa forma. A indução perfeita (prova exaustiva) se dá ao fazer a tabela verdade da igualdade, verificando se os valores literais realmente coincidem. Quando a expressão começa a ter muitas variáveis e/ou se tornar muito grande, esse método pode se tornar inviável devido a grande quantidade de estados possíveis que o sistema pode ter.
- ▶ manipulação algébrica: Levando em consideração os axiomas e teoremas é possível manipular as igualdades de forma que a igualdade seja provada.
- ▶ diagrama de Ven: *Nos próximos capítulos.*

Provas de expressões na álgebra de Boole com diagrama de Venn

Se interpretarmos **AND** ou \cdot como \cap e **OR** ou $+$ como \cup é possível fazer a prova de uma igualdade usando o diagrama de Venn.

Para começar, uma dica interessante é representar as relações do espaço para identificar cada uma das regiões de acordo com a expressão avaliada.

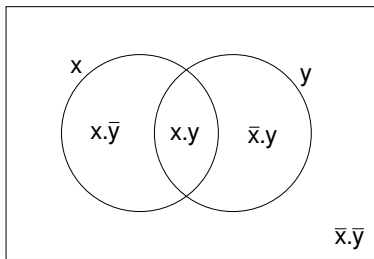


Figura: Relações de um espaço com duas variáveis

A partir disso é possível escrever outras relações:

$$\begin{aligned}(x.\bar{y}) + (x.y) &= x \\ (\bar{x}.y) + (x.y) &= y \\ (x.\bar{y}) + (x.y) + (\bar{x}.y) &= x + y\end{aligned}\tag{7}$$

Provas de expressões na álgebra de Boole com diagrama de Venn

Como exemplo provaremos o do teorema de DeMorgan: $\overline{(x \cdot y)} = \bar{x} + \bar{y}$

Lado esquerdo da igualdade.

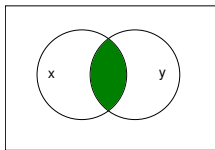


Figura: $(x \cdot y)$

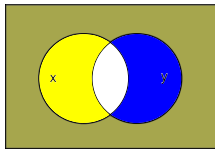


Figura: $\overline{(x \cdot y)}$

Lado direito da igualdade.

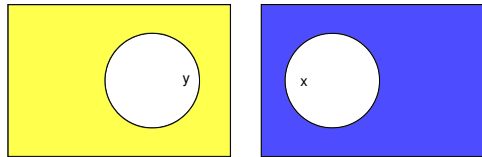


Figura: \bar{x} e \bar{y}

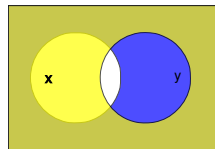


Figura: $\bar{x} + \bar{y}$

Provas de expressões na álgebra de Boole com diagrama de Venn

Como exemplo provaremos $(x.y) + (y.z) + (\bar{x}.z) = (x.y) + (\bar{x}.z) \Leftrightarrow (x + y).(y + z).(\bar{x} + z) = (x + y).(\bar{x} + z)$ consenso (sumiço :D)

Lado esquerdo da igualdade.

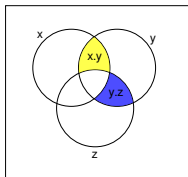


Figura: $(x.y) + (y.z)$

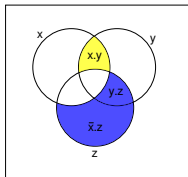


Figura: $(x.y) + (y.z) + (\bar{x}.z)$

Lado direito da igualdade.

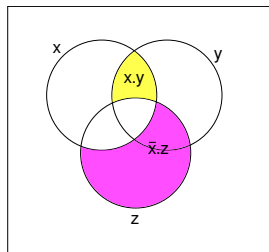


Figura: $(x.y) + (\bar{x}.z)$

A partir dos teoremas e propriedades apresentados mostre que $(x + y).(x + z) = x + y.z$.
Indique quais os teoremas e/ou propriedades usados na solução.

Crie dualidades para

- ▶ 0.0
- ▶ 1.1
- ▶ $x = 0 \implies \bar{x} = 0$

Até agora, revisamos a lógica, as tabelas-verdade e a álgebra booleana. No entanto, ainda não abordamos os circuitos digitais propriamente ditos. A partir de agora, exploraremos a correlação entre as expressões lógicas da álgebra booleana e o projeto de circuitos elétricos.

Circuitos combinacionais

A chave binária é um componente que permite ou impede a passagem da corrente elétrica dado seu estado. A partir dela criaremos circuitos mais complexos.

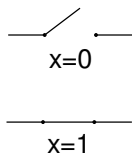


Figura: Chave binária

A chave binária pode ser representada de outra forma:

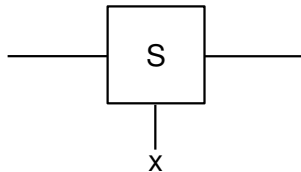
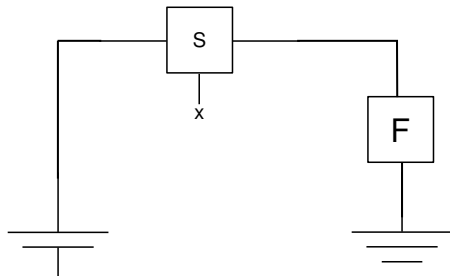


Figura: Chave binária

Circuitos combinacionais

Agora considere uma função que depende um argumento x para definir seu estado. Tal função pode ser representada no circuito abaixo:



$$F = 1 \forall x = 1$$

$$F = 0 \forall x = 0$$

Portanto F é uma função lógica de **uma** variável tal que $F(x) = x$

Figura: Função lógica de uma variável. No circuito F pode ser ligada a qualquer elemento de saída com um led ou outro circuito.

Circuitos combinacionais

A partir de agora é possível construir circuitos mais complexos como **OR** e **AND**.

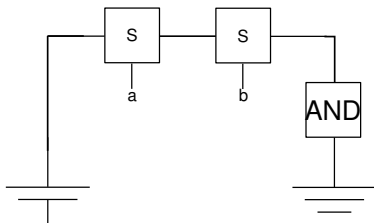


Figura: Circuito **AND**: a e b são suas entradas. Aqui se pode constatar que este circuito pode receber de 2 a n entradas.

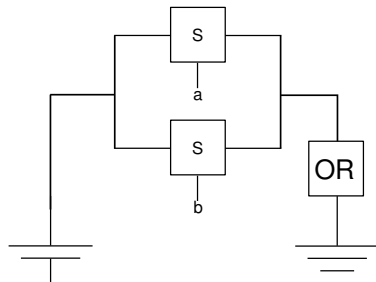


Figura: Circuito **OR**: a e b são suas entradas. Aqui se pode constatar que este circuito pode receber de 2 a n entradas.

Circuito **NOT**

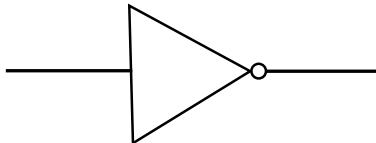


Figura: Circuito **NOT**: Inverte a entrada a .

Nada aqui por questões estéticas...

Circuitos combinacionais

E assim como vimos na álgebra de Boole **OR** e **AND** podem ser combinados.

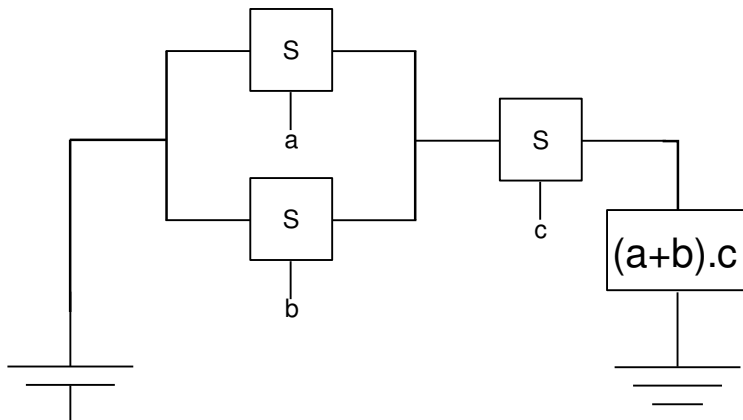


Figura: O **OR** recebe as entradas a e b , o circuito **AND** recebe c e a saída de **OR**

Para representar circuitos simples os símbolos já vistos são suficientes, porém, quando a complexidade aumenta pode ficar beeeeeemmmm complicado entender o que se passa. Então precisamos aumentar o nível de abstração encapsulando esse componentes.

Circuitos combinacionais

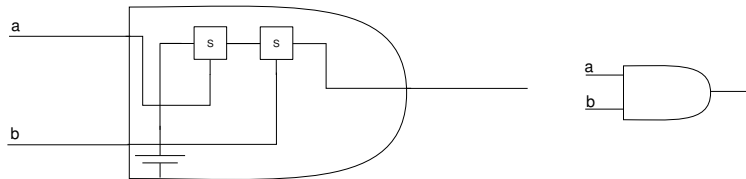


Figura: Abstração de circuito **AND** para a porta **AND**

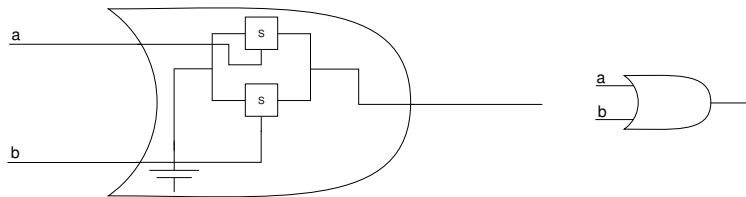


Figura: Abstração de circuito **OR** para a porta **OR**

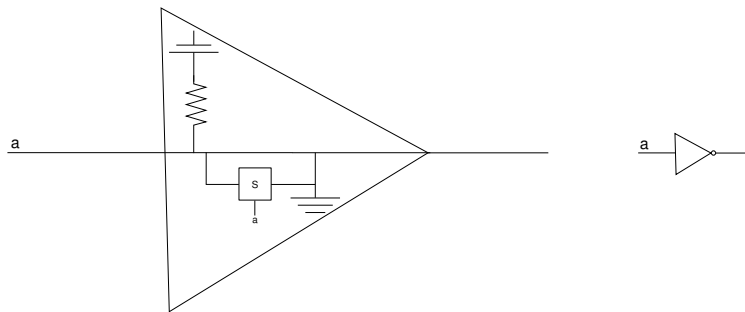


Figura: Abstração de circuito **NOT** para a porta **NOT**

Veja como fica o circuito $(a + b).c$ com as novas abstrações.

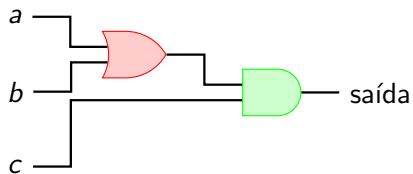


Figura: Circuito correspondente a expressão lógica $(a+b).c$