UNIDADE 1 – LÓGICA DAS PROPOSIÇÕES E A APLICAÇÃO EM CIRCUITOS LÓGICOS E ALGORITMO

MÓDULO 1 – A RELAÇÃO ENTRE PROPOSIÇÕES E EXPRESSÕES LÓGICAS

01

1 - TRANSFORMAÇÃO DA PROPOSIÇÃO EM EXPRESSÃO LÓGICA

Quanto mais reduzida e simplificada é uma proposição lógica, melhor será para estudá-la. Principalmente, quando se aprende que o desenvolvimento de circuitos lógicos digitais, um importante fundamento computacional, também é consequentemente facilitado pela simplificação e redução de conectivos.

Para entender tal relação, basta compreender que os conectivos e as operações da lógica matemática correspondem às chamadas portas lógicas dos circuitos lógicos digitais, que, por sua vez, integram a arquitetura dos computadores.

A correspondência entre tal segmento da arquitetura de computadores com a lógica matemática tem inicio na relação existente entre uma Proposição Lógica e uma Expressão Lógica, conforme detalhamento a seguir.

Proposição Lógica e Expressão Lógica
São elementos correspondentes!

02

O primeiro passo para transitarmos por esta "ponte" entre a arquitetura de computadores e a lógica matemática é a transformação de uma proposição lógica em sua correspondente expressão lógica.

Proposições Lógicas				Expressões Lógicas		
Nome da Operação	Linguagem Coloquial	Linguagem Simbólica		Nome da Operação	Representação simbólica	
Conjunção	"e"	$p \wedge q$	→	AND	P•Q	*Produto*
Disjunção	"ou"	$p \vee q$	→	OR	P+Q	*Soma*
Disjunção exclusiva	"ou...ou"	$p \veebar q$	→	XOR	P⊕Q	*Soma exclusiva*
Negação	"não"	$\sim p$	→	NOT	$\overline{P}$	*Inversora*

03

Para encontrar a expressão lógica correspondente da condicional e da bicondicional, vamos aplicar as regras de simplificação para obter suas equivalentes na forma normal (FN):

Proposições Lógicas				Expressões Lógicas
Nome da Operação	Linguagem Coloquial	Linguagem Simbólica	Simplicação à FN	Representação simbólica
Condicional	"Se-então"	$p \rightarrow q$	$\sim p \vee q$	$\bar{P} + Q$
Bicondicional	"se somente se"	$p \leftrightarrow q$	$(\sim p \vee q) \wedge (\sim q \vee p)$	$(\bar{P} + Q) \bullet (\bar{Q} + P)$

Dessa forma, qualquer proposição pode ser transformada em sua correspondente expressão lógica, bastando substituir os sinais pela respectiva representação simbólica. Por exemplo, considere a Proposição:

$$\sim(p \wedge q)$$

Sua Expressão Lógica correspondente é obtida pelos seguintes passos:

1º) Substituição do símbolo dentro do parêntese:

$$\sim(P \bullet Q)$$

2º) Se a negação está atuando no parêntese todo e não apenas em uma proposição simples qualquer, então a "barra" inversora da expressão lógica deverá ficar sobre o parêntese inteiro:

$$\overline{(P \bullet Q)}$$

Esta expressão lógica é também chamada de NAND, que significa a Negação da operação AND.

Resumindo, fizemos a seguinte transformação:

Proposição Lógica	Expressão Lógica	
Linguagem simbólica	Nome da Operação	Representação simbólica
$\sim(p \wedge q)$	NAND	$\overline{(P \bullet Q)}$

04

De forma similar, a proposição abaixo origina a operação NOR no projeto de circuitos lógicos:

$$\sim(p \vee q)$$

A expressão lógica correspondente é obtida da mesma forma, pelos seguintes passos:

1º) Substituição do símbolo dentro do parêntese:

~(P+Q)

2º) Se a negação está atuando no parêntese todo e não apenas em uma proposição simples qualquer, então a "barra" inversora da expressão lógica deverá ficar sobre o parêntese inteiro:

$$\overline{(P+Q)}$$

Assim, obtém-se a Negação da operação OR, denominada de NOR na notação dos circuitos lógicos digitais.

05

Veja então a transformação que foi feita:

Proposição Lógica	Expressão Lógica	
Linguagem simbólica	Nome da Operação	Representação simbólica
$\sim(p \vee q)$	NOR	$\overline{(P+Q)}$

Neste momento vamos relembrar a Regra de Simplificação de DE MORGAN. Acompanhe a seguir a sua aplicação sobre as expressões lógicas recentemente aprendidas, a NAND e a NOR:

$\sim(p \wedge q) \Leftrightarrow (\sim p \vee \sim q)$

Ou seja:

$\overline{(P \bullet Q)} \Leftrightarrow \overline{P}+\overline{Q}$

$\sim(p \vee q) \Leftrightarrow (\sim p \wedge \sim q)$

Ou seja:

$\overline{P+Q} \Leftrightarrow (\overline{P} \bullet \overline{Q})$

De Morgan
A regra é a comprovação das seguintes equivalências: $\sim(p \wedge q) \Leftrightarrow (\sim p \vee \sim q)$ e $\sim(p \vee q) \Leftrightarrow (\sim p \wedge \sim q)$ e. Ou seja, desloca-se a negação de fora do parêntese para dentro, deixando-a incidente sobre cada uma das partes, invertendo o conectivo central de $\wedge$ para $\vee$, e vice-versa.

06

De forma similar ao caso da porta OR, vamos agora ver que existe também a negação da XOR, mas com algumas particularidades explicadas a seguir:

A partir da proposição:

~ (pVq)

Obtém-se a expressão correspondente:

$\overline{P \oplus Q}$

No entanto, a negação da XOR não ganha um nome próprio, como, por exemplo, ocorreu com a negação da AND que tem a denominação de NAND.

Há também outra distinção:

Já que não há regra de DE MORGAN sobre v, lembre-se também que não haverá DE MORGAN quando dentro do parêntese da expressão for a porta $\oplus$.

Assim, caso se queira reduzir a expressão $\overline{P \oplus Q}$ à Forma Normal (FN), deve-se aplicar outra regra de simplificação. Veja a seguir:

$$P \oplus Q \Leftrightarrow (\overline{P} \bullet Q) + (P \bullet \overline{Q})$$

07

Note que a proposição $(\overline{P} \bullet Q) + (P \bullet \overline{Q})$ garante a regra da disjunção exclusiva, devido ao seguinte:

Quando a primeira parte $(\overline{P} \bullet Q)$ for Verdade, a segunda parte $(P \bullet \overline{Q})$ será obrigatoriamente Falsa, e vice-versa. Assim, resulta em Verdade a disjunção entre a primeira e a segunda parte, $(\overline{P} \bullet Q) + (P \bullet \overline{Q})$, somente quando P e Q tiverem valores lógicos distintos, conforme define exatamente a regra da disjunção exclusiva.

A comprovação de tal equivalência é demonstrada a seguir, via tabela verdade:

P	Q	$\overline{P}$	$(\overline{P} \bullet Q)$	$\overline{Q}$	$(P \bullet \overline{Q})$	$(\overline{P} \bullet Q) + (P \bullet \overline{Q})$	$\Leftrightarrow$	$(P \oplus Q)$
V	V	F	F	F	F	F		V
V	F	F	F	V	V	V		V
F	V	V	V	F	F	V		V
F	F	V	F	V	F	F		F

08

Comprovada então a equivalência $P \oplus Q \Leftrightarrow (\overline{P} \bullet Q) + (P \bullet \overline{Q})$, pode-se retornar ao objetivo de simplificar a expressão $\overline{P \oplus Q}$ à sua FN. Vejamos:

Proposição a ser simplificada à sua FN é:

$$\overline{P \oplus Q}$$

Aplicando a equivalência $P \oplus Q \Leftrightarrow (\overline{P} \bullet Q) + (P \bullet \overline{Q})$, tem-se:

$$\overline{(\overline{P} \bullet Q) + (P \bullet \overline{Q})}$$

Aplicando então DE MORGAN, resulta em:

$$\overline{(\overline{P} \bullet Q)} \bullet \overline{(P \bullet \overline{Q})}$$

Aplicando DE MORGAN, em cada um dos parênteses, obtém-se:

$$(\overline{\overline{P}} + \overline{Q}) \bullet (\overline{P} + \overline{\overline{Q}})$$

Aplicando a Dupla Negação, tem-se a FN, que é especificamente uma FNC – Forma Normal Conjuntiva:

$$(P + \overline{Q}) \bullet (\overline{P} + Q)$$

Perceba que a notação de expressão lógica utiliza letras maiúsculas na substituição das proposições simples. Esta é uma prática da área, mas não estaria errado manter o uso de letras minúsculas.

09

2 - TABELA VERDADE DE EXPRESSÕES LÓGICAS

A representação dos valores lógicos alimentados na tabela verdade é diferenciada entre o estudo da lógica das proposições e o estudo dos circuitos lógicos. Ao invés de V e F, a prática é usar, respectivamente, 1(um) e 0 (zero), conforme demonstrado a seguir para a operação AND:

P	Q	P • Q
1	1	1
1	0	0
0	1	0
0	0	0

Há também o hábito de iniciar o preenchimento das primeiras colunas por 0 (zero):

P	Q	P • Q
0	0	0
0	1	0
1	0	0
1	1	1

10

Ressalta-se, porém, que o resultado final será o mesmo, independente do preenchimento, iniciar por 1 ou por 0, pois a única diferença será a ordem com que as combinações serão analisadas. Vejamos em detalhe a seguir:

Tabela verdade com o preenchimento iniciado por 1:

	P	Q	P • Q
1.	1	1	1
2.	1	0	0
3.	0	1	0
4.	0	0	0

Tabela verdade com o preenchimento iniciado por 0:

	P	Q	P • Q
1.	0	0	0
2.	0	1	0
3.	1	0	0
4.	1	1	1

Note que o resultado final é igual entre as duas tabelas acima. Quando as linhas são comparadas, conforme ilustram os destaques feitos com cores, os mesmos valores são encontrados em ambas. Esta conclusão é válida para todas as tabelas verdades.

11

Para finalizar, veja abaixo, os resultados das tabelas verdades das portas AND, OR e XOR alimentadas pelos valores 1 e 0, iniciando-se por 0, devido à referida prática dos circuitos lógicos:

A	B	AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Os valores “1” e “0” significam, respectivamente, existência e inexistência de eletricidade, ou seja, corrente elétrica “ligada” e “desligada” no circuito digital, conforme é explicado no próximo item.

3 - Representação Gráfica de Expressões Lógicas

As operações AND, OR, XOR e NOT são também denominadas como Portas Lógicas e possuem as seguintes representações gráficas utilizadas no projeto de montagem de circuitos lógicos:

Operações Lógicas			Portas Lógicas	
Nome da Operação	Linguagem Coloquial	Linguagem Simbólica	Nome da Porta	Representação Gráfica
Conjunção	“e”	$p \wedge q$	AND	
Disjunção	“ou”	$p \vee q$	OR	
Disjunção exclusiva	“ou...ou”	$p \veebar q$	XOR	
Negação	“não”	$\sim p$	NOT	

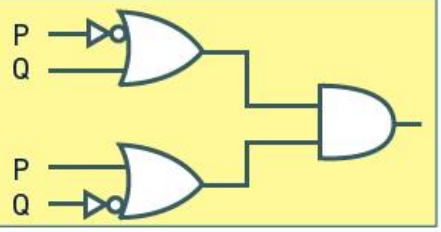
Além dessas portas lógicas, há também as representações de NAND e NOR:

Portas Lógicas	
Nome da Porta	Representação Gráfica
NAND	
NOR	

13

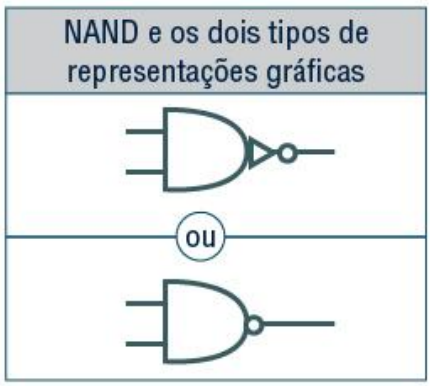
Para as operações condicional e bicondicional, vimos anteriormente que não há correspondência direta entre a lógica das proposições e uma nomenclatura de portas lógicas. Sendo assim, também não haverá uma correspondência direta quanto à representação gráfica.

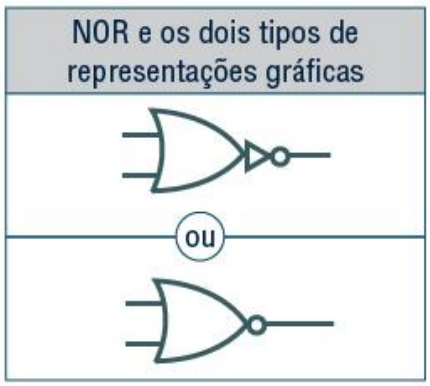
Dessa forma, quando for necessário converter uma proposição $p \rightarrow q$ para a linguagem de circuitos lógicos, aplicam-se antes as regras de simplificações cabíveis. Veja a seguir:

Condicional	$p \rightarrow q$	$\sim p \vee q$	$\overline{P} + Q$	
Bicondicional	$p \leftrightarrow q$	$(\sim p \vee q) \wedge (\sim q \vee p)$	$(\overline{P} + Q) \bullet (\overline{Q} + P)$	

14

Para facilitar a representação gráfica da negação (NOT) pode usar apenas o círculo ○ ao invés de .
Por exemplo:





15

Para exemplificar a integração da notação de circuito lógico com a tabela verdade, mostra-se abaixo um circuito simples, composto apenas pela porta OR, acompanhado da respectiva tabela verdade:

Entradas		Saída
A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

Quanto à ilustração acima, é importante saber:

• A: chamada de variável de entrada do circuito lógico (ou chamada simplesmente de Entrada do circuito).

• B: também é uma entrada do circuito lógico. As variáveis de entrada correspondem ao conceito de proposições simples na lógica das proposições.

• O resultado final da tabela verdade é chamado de Saída do circuito.

Assim, a leitura do funcionamento desde circuito é feita da seguinte maneira:

• Na primeira linha da tabela as entradas A e B assumem os valores 0 e 0, cuja combinação A OR B obtém o valor 0 como resultado de saída do circuito.

• Na segunda linha, a combinação das entradas é 0 e 1, sendo 1 o respectivo valor de saída.

• Na terceira e na quarta linhas as saídas também são 1, tanto para a entrada 1 e 0, como para a entrada 1 e 1.

16

A leitura e entendimento de circuitos lógicos dependem do aprendizado do funcionamento das portas lógicas, que, conforme vimos, possuem relação com o raciocínio e regras das operações da lógica das proposições.

Neste ponto do estudo, o necessário está no aprendizado das portas lógicas, resumidas no quadro a seguir:

Nome	Símbolo	Entradas		Saídas
		A	B	A or B
E (AND)	 $F = A \cdot B$	0	0	0
		0	1	0
		1	0	0
		1	1	1
OU (OR)	 $F = A + B$	0	0	0
		0	1	1
		1	0	1
		1	1	1
NÃO (NOT) INVERSORA	 $F = \overline{A}$	-	-	-
		0	-	1
		1	-	0
		-	-	-
NÃO E (NAND)	 $F = \overline{A \cdot B}$	0	0	1
		0	1	1
		1	0	1
		1	1	0
NÃO OU (NOR)	 $F = \overline{A + B}$	0	0	1
		0	1	0
		1	0	0
		1	1	0
OU EXCLUSIVO (XOR)	 $F = A \oplus B = \overline{A}B + A\overline{B}$	0	0	0
		0	1	1
		1	0	1
		1	1	0

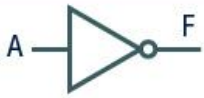
17

RESUMO

Ao final desta Unidade será completa a compreensão da aplicação da Lógica das Proposições no desenvolvimento de circuitos lógicos digitais, que, por sua vez, é um elemento relevante ao estudo da arquitetura dos computadores.

O primeiro passo é compreender que os conectivos/operações das proposições lógicas correspondem às chamadas portas lógicas das expressões lógicas que representam os circuitos digitais.

Sendo assim, a correspondência com tal segmento da arquitetura de computadores é iniciada pela relação existente entre Proposição Lógica e Expressão Lógica, conforme consolidado a seguir:

Nome	Símbolo	Entradas		Saídas
		A	B	A or B
E (AND)	 $F = A \cdot B$	0	0	0
		0	1	0
		1	0	0
		1	1	1
OU (OR)	 $F = A + B$	0	0	0
		0	1	1
		1	0	1
		1	1	1
NÃO (NOT) INVERSORA	 $F = \overline{A}$	-	-	-
		0	-	1
		1	-	0
		-	-	-
NÃO E (NAND)	 $F = \overline{A \cdot B}$	0	0	1
		0	1	1
		1	0	1
		1	1	0
NÃO OU (NOR)	 $F = \overline{A + B}$	0	0	1
		0	1	0
		1	0	0
		1	1	0
OU EXCLUSIVO (XOR)	 $F = A \oplus B = \overline{A}B + A\overline{B}$	0	0	0
		0	1	1
		1	0	1
		1	1	0

No quadro acima está também presente a aplicação do estudo de tabela verdade, uma importante ferramenta de interpretação de valores lógicos, que agora sabemos que é usada não apenas na análise de sentenças e pensamentos humanos, mas também na análise e validação de circuitos lógicos digitais.

UNIDADE 1 – LÓGICA DAS PROPOSIÇÕES E A APLICAÇÃO EM CIRCUITOS LÓGICOS E ALGORITMO

MÓDULO 2 – A LÓGICA DAS PROPOSIÇÕES E OS CIRCUITOS LÓGICOS

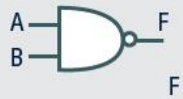
01

1 - MONTAGEM DE CIRCUITOS LÓGICOS

O primeiro passo foi alcançado! Já foi compreendido que os conectivos/operações da lógica das proposições possuem correspondência com as portas lógicas dos circuitos lógicos digitais.

Sendo assim, sabe-se agora que simplificar e reduzir conectivos viabiliza o desenvolvimento de circuitos lógicos digitais, um importante fundamento computacional.

Vimos que a montagem e a leitura de circuitos lógicos dependem do aprendizado do funcionamento das seguintes portas lógicas:

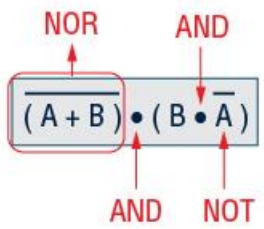
Nome	Símbolo	Entradas		Saída
		A	B	A or B
E (AND)	 $F = A \cdot B$	0	0	0
		0	1	0
		1	0	0
		1	1	1
OU (OR)	 $F = A + B$	0	0	0
		0	1	1
		1	0	1
		1	1	1
NÃO (NOT) INVERSORA	 $F = \overline{A}$	-	-	-
		0	-	1
		1	-	0
		-	-	-
NÃO E (NAND)	 $F = \overline{A \cdot B}$	0	0	1
		0	1	1
		1	0	1
		1	1	0
NÃO OU (NOR)	 $F = \overline{A + B}$	0	0	1
		0	1	0
		1	0	0
		1	1	0
OU EXCLUSIVO (XOR)	 $F = A \oplus B = \overline{A}B + A\overline{B}$	0	0	0
		0	1	1
		1	0	1
		1	1	0

02

Sendo conhecidas as grafias das portas lógicas, torna-se possível desenhar o projeto de qualquer circuito lógico digital. Considere então a seguinte expressão lógica:

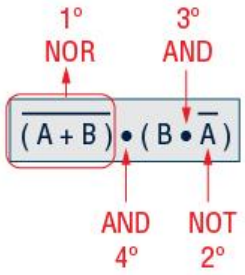
$$\overline{(A + B)} \cdot (B \cdot \overline{A})$$

Analisando as portas lógicas, temos:

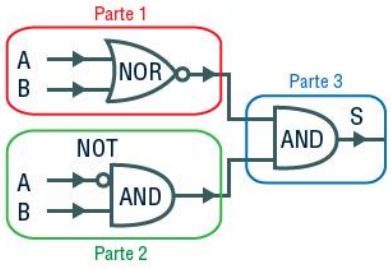


03

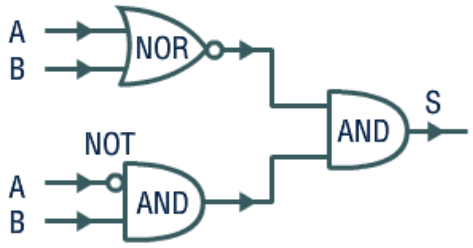
Para a construção do respectivo circuito lógico, deve-se atentar para a mesma ordem de precedência estudada na lógica das proposições. Assim, o projeto deste circuito deve ser montado na seguinte ordem:



Monta-se então o circuito por partes, obedecendo a ordem expressada:



Por fim, tem-se o seguinte circuito lógico, onde A e B são as entradas e S é a saída do resultado final:



04

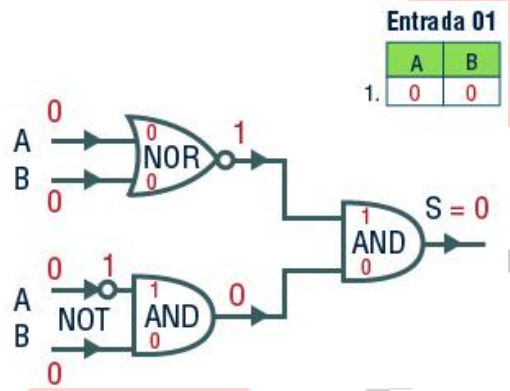
A ordem de precedência é também importante para que o projeto do circuito garanta que o funcionamento correto, que na prática é representado pelo caminho percorrido pela energia elétrica que alimenta o circuito. Para tanto, basta alimentar as variáveis A e B com as combinações dos valores “1” e “0”, que são, respectivamente, corrente “ligada” e “desligada”.

As referidas combinações dos valores 1” e “0” são obtidas das primeiras colunas da tabela verdade, que no caso da expressão $(\overline{A+B}) \bullet (B \bullet \overline{A})$ são as seguintes:

	A	B	
Linha 1.	0	0	← Combinação 1
Linha 2.	0	1	← Combinação 2
Linha 3.	1	0	← Combinação 3
Linha 4.	1	1	← Combinação 4

05

Alimentando a entrada do circuito lógico com a combinação da linha 1, o percurso da corrente elétrica será:

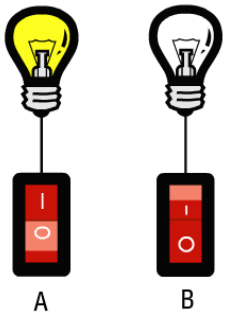


Percorrido o circuito inteiro, o resultado final foi 0 (zero), conforme ilustrado por S = 0. Isto significa que ao final do circuito não haverá a presença de corrente elétrica quando as entradas A e B estiverem desligadas (combinação de entrada A=0 e B=0).

06

Para melhorar a compreensão, imagine que as entradas A e B são interruptores elétricos que ligam e desligam suas respectivas lâmpadas. Ao ligar o interruptor A é acesa a lâmpada A, confirmando a presença de corrente elétrica, o que significa que o valor de entrada de A recebeu o valor 1.

Da mesma forma, ao desligar o interruptor A, apaga-se a respectiva lâmpada A, representando a inexistência de corrente elétrica, devido à alteração do valor da variável de entrada para 0.

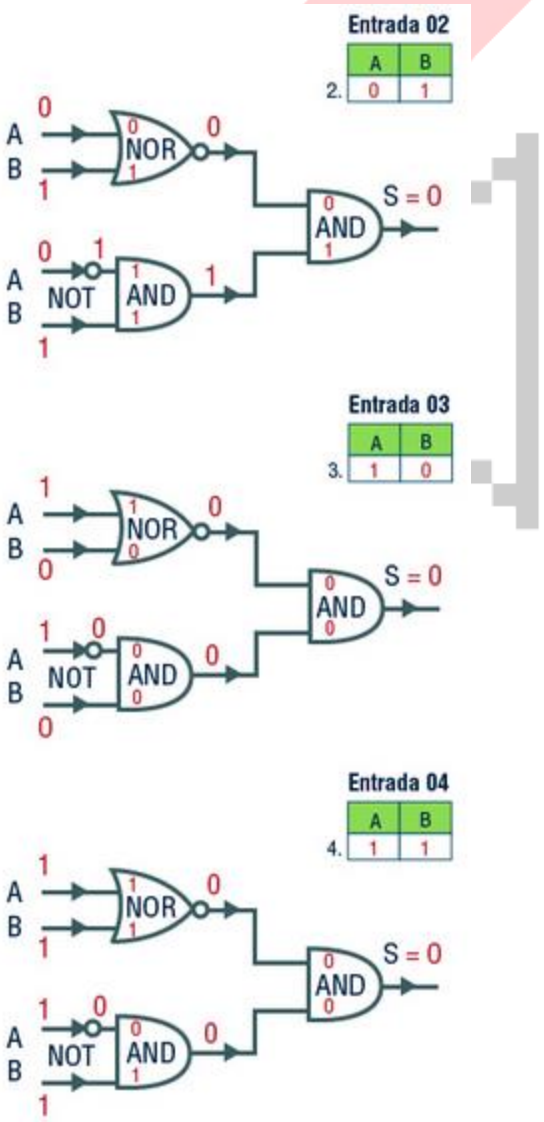


Apoiados nesta analogia, vamos descrever a seguir a conclusão referente ao circuito lógico de $\overline{(A + B)} \cdot (B \cdot \overline{A})$:

Quando a combinação de entrada for A=0 e B=0, o resultado final será uma “lâmpada apagada”, pois foi obtida a saída=0. Ou seja, para este circuito, quando as duas entradas não forem alimentadas com corrente elétrica, então a saída também não receberá corrente elétrica.

07

Lembre que este foi o resultado obtido após a aplicação das entradas A=0 e B=0, correspondentes à combinação da primeira linha da tabela verdade. Mas, ainda resta a dúvida: Quais serão os resultados de saída após aplicar ao circuito as demais combinações de entrada? Veja a seguir.



Ou seja, para este circuito da expressão $\overline{(A + B)} \cdot (B \cdot \overline{A})$, ainda que uma das entradas receba corrente elétrica, a saída permanece sem receber corrente elétrica, mantendo, portanto, o estado "lâmpada apagada" obtido na combinação 1. Até mesmo quando se alimente com corrente elétrica as duas entradas A e B, o resultado final não é alterado, como foi o caso ilustrado pela combinação 4.

08

2 - Validação do funcionamento do circuito lógico via tabela verdade

O funcionamento de um circuito lógico pode, e normalmente é, validado por meio de tabela verdade, sendo possível confirmar os resultados intermediários produzidos ao longo do percurso do circuito, até a obtenção do respectivo resultado final.

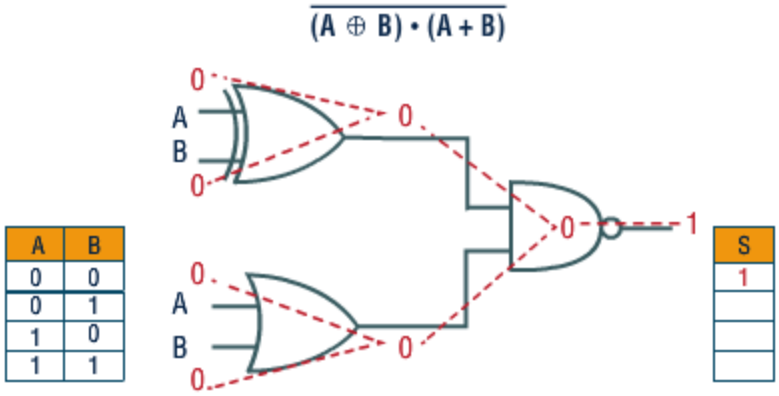
Retomando o exemplo do circuito da expressão $\overline{(A + B)} \cdot (B \cdot \overline{A})$, percebe-se que estamos trabalhando com uma Contradição!

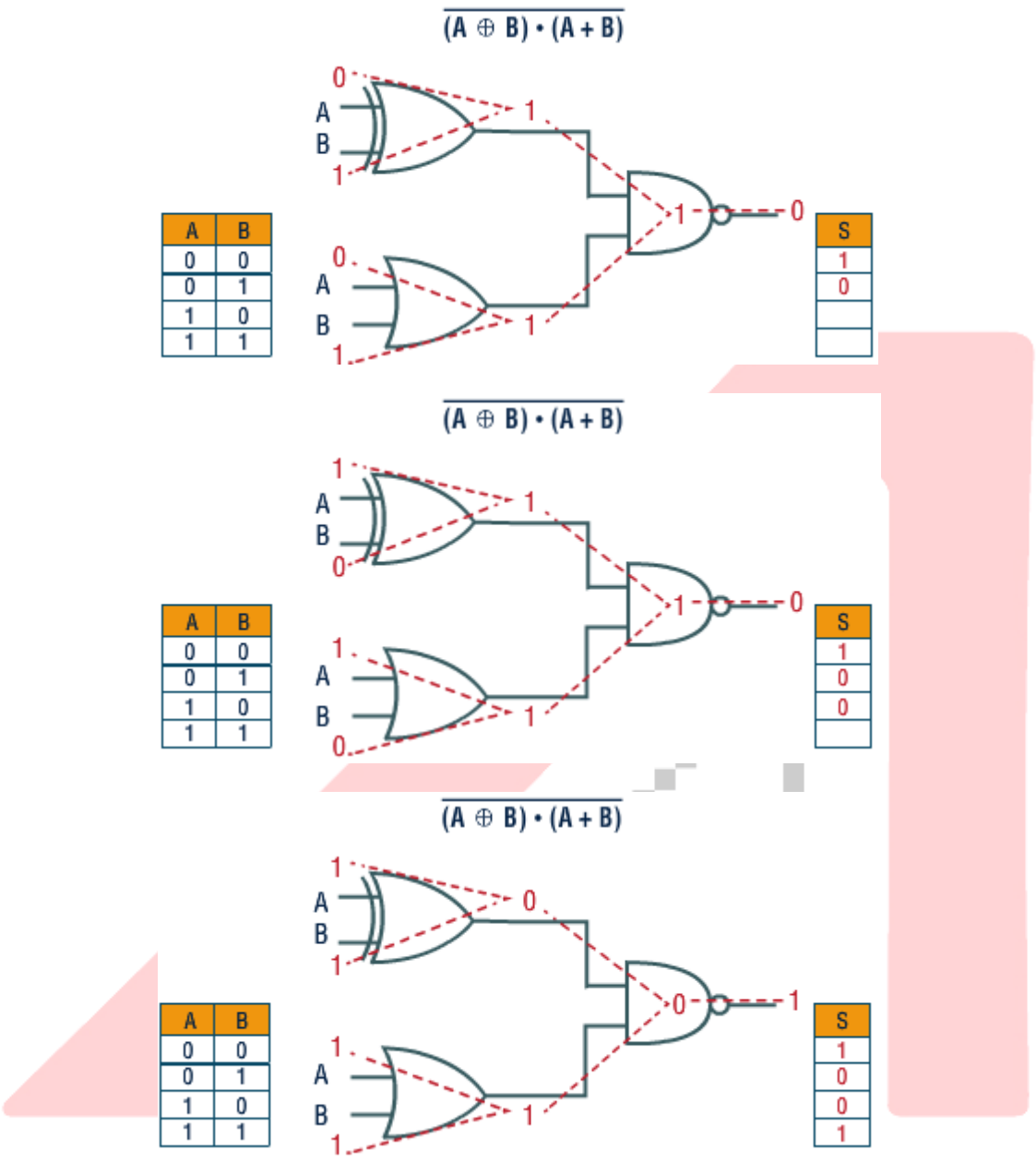
	A	B	(A + B)	$\overline{(A+B)}$	$\overline{A}$	$B \bullet \overline{A}$	$\overline{(A+B)} \bullet (B \bullet \overline{A})$
1.	0	0	0	1	1	0	0
2.	0	1	1	0	1	1	0
3.	1	0	1	0	0	0	0
4.	1	1	1	0	0	0	0

A constatação da Contradição confirma a total ausência de eletricidade das saídas obtidas na montagem do circuito lógico, comprovando que todas as combinações de entradas geram sempre a saída igual a 0. Assim, atestada a equivalência de entre os resultados do circuito lógico e da tabela verdade, valida-se o projeto concebido para $\overline{(A + B)} \cdot (B \cdot \overline{A})$.

09

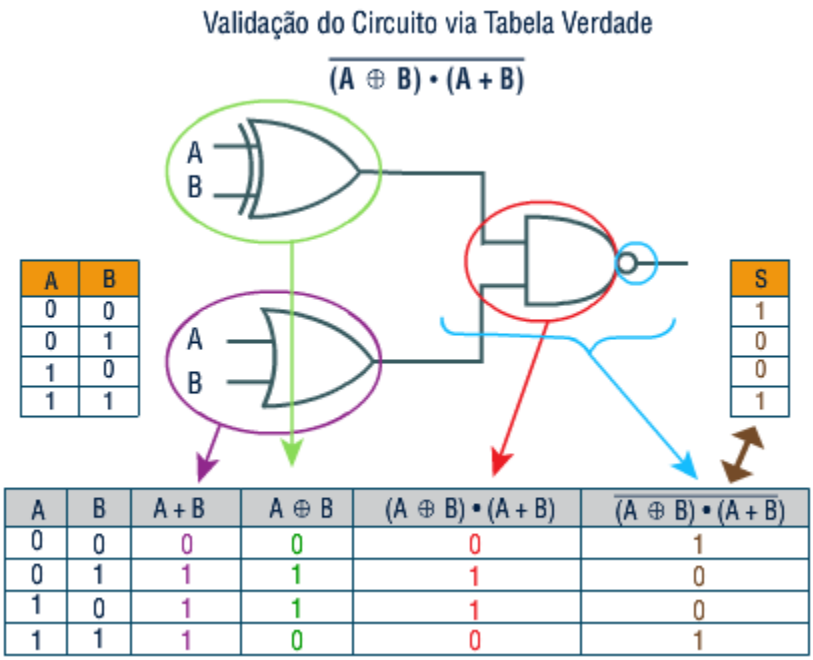
Para consolidar o estudo, vamos agora fazer construir e testar o circuito lógico da expressão abaixo:





10

A validação do circuito pode ser evidenciada, conforme demonstrado a seguir, devido à equivalência entre os resultados de Saída do circuito e o resultado da tabela verdade. Vale ressaltar que esta validação não é obrigatória na prática, mas possui efeito didático relevante, proporcionando a visão integrada do estudo.



A	B	A + B	A ⊕ B	(A ⊕ B) • (A + B)	$\overline{(A \oplus B) \bullet (A + B)}$
0	0	0	0	0	1
0	1	1	1	1	0
1	0	1	1	1	0
1	1	1	0	0	1

11

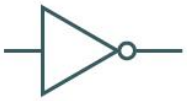
RESUMO

Quando se aprende que o desenvolvimento de circuitos lógicos digitais, é facilitado e viabilizado pela simplificação e redução de conectivos lógicos, aprende-se então uma importante aplicação do estudo da Lógica da área computacional.

Sem o entendimento desta relação ficaria complicado compreender a aplicação de conceitos estudados, tais como FN, FNC, FND, Equivalências notáveis e Tabela verdade.

A correspondência com tal segmento da arquitetura de computadores está apoiada nos seguintes fundamentos básicos:

BLOCOS LÓGICOS BÁSICOS

PORTA	Simbologia	Tabela da Verdade	Função Lógica	Espressão
E (AND)		A B S 0 0 0 0 1 0 1 0 0 1 1 1	**Função E:** Assume 1 quando todas as variáveis forem 1 e 0 nos outros casos.	**S = A.B**
OU (OR)		A B S 0 0 0 0 1 1 1 0 1 1 1 1	**Função OU:** Assume 0 quando todas as variáveis forem 0 e 1 nos outros casos.	**S = A+B**
NÃO (NOT)		A S 0 1 1 0	**Função NÃO:** Inverte a variável aplicada à sua entrada.	**S = $\bar{A}$**
NÃO E (NAND)		A B S 0 0 1 0 1 1 1 0 1 1 1 0	**Função NAND:** Inverso da função E.	**S =$\overline{(A.B)}$**
NÃO OU (NOR)		A B S 0 0 0 1 0 1 0 0 1 1 0	**Função NOR:** Inverso da função OU.	**S = $\overline{(A+B)}$**
OU EXCLUSIVO (XOR)		A B S 0 0 0 0 1 1 1 0 1 1 1 0	**Função OU Exclusivo:** Assume 1 quando as variáveis assumirem valores diferentes entre si.	**S = A⊕B** **S = $\bar{A}$.B + A.$\bar{B}$**

Parabéns! Você não somente compreendeu o raciocínio da lógica das proposições, como também a sua aplicação em circuitos lógicos digitais! Este é um aprendizado importante e útil.

UNIDADE 1 – LÓGICA DAS PROPOSIÇÕES E A APLICAÇÃO EM CIRCUITOS LÓGICOS E ALGORITMO

MÓDULO 3 – ESTRUTURAS ALGORÍTMICAS E A LÓGICA DAS PROPOSIÇÕES

01

1 - ESTRUTURAS DE DECISÃO E A LÓGICA DAS PROPOSIÇÕES

A lógica booleana é também base do raciocínio das estruturas algorítmicas, que, por sua vez, são as células da fase de desenvolvimento dos *softwares* computacionais.

A seguir são abordadas as convergências de interpretações entre as regras das operações lógicas e as regras das estruturas algorítmicas, conforme detalhamento dos dois grandes grupos a seguir:

• estruturas algorítmicas de decisão e
• estruturas algorítmicas de repetição.

A lógica do raciocínio das operações condicional e bicondicional é correspondente à utilizada pelas estruturas algorítmicas de decisão, que atribuem ao programa a capacidade de decidir entre duas ou mais possibilidades de execução.

Perceba que a interpretação das estruturas de decisão parte da análise de uma **condição**. Relacionando com a lógica das proposições, uma condição tem que ser um pensamento completo, pronto para ser julgado como verdadeiro ou falso, conforme define o conceito de proposição lógica.

Além disso, os princípios da lógica das proposições são também preservados na interpretação da condição: uma condição ou é verdadeira ou é falsa, sendo excluída a possibilidade de um terceiro valor, além de não poder ser verdadeira e falsa ao mesmo tempo.

REGRAS DAS OPERAÇÕES LÓGICAS

Operação Lógica			Quando é verdade?
$p \wedge q$	"e"	Conjunção	Quando ambos são verdade.
$p \vee q$	"ou"	Disjunção	Quando pelo menos um dos dois for verdade.
$p \veebar q$	"ou...ou"	Disjunção exclusiva	Quando apenas um dos dois é verdade.
$p \rightarrow q$	"Se-então"	Condicional	Só é falso quando V→F. É verdade nos demais casos.
$p \leftrightarrow q$	"se somente se"	Bicondicional	Quando **p** e **q** possuem valores lógicos iguais.
~p	"não"	Negação	Quando **p** é falso.

02

Todas as estruturas de decisão são também compostas por um ou mais **comandos** com execução dependente do julgamento da condição. Assim, o programa executará o comando ou os comandos previstos, se e somente se a condição for verdadeira, uma relação clara com a bicondicional.

	p	q	$p \leftrightarrow q$
1.	V	V	V
2.	V	F	F
3.	F	V	F
4.	F	F	V

A relação das estruturas de decisões com uma proposição condicional é restrita às linhas em que o valor de **p** é verdade, conforme destacado abaixo, pois os comandos de dentro da estrutura algorítmica somente são executados quando a condição for satisfeita, ou seja, quando o seu valor for uma verdade.

	p	q	p → q
1.	V	V	V
2.	V	F	F
3.	F	V	V
4.	F	F	V

Ao fazer a seguinte correspondência fica mais fácil o entendimento:

t → q
onde t é uma tautologia.

Aplicando as regras de simplificação em t → q, o resultado final será sempre o próprio q:

	p	q	p → q
1.	V	V	V
2.	V	F	F
3.	F	V	V
4.	F	F	V

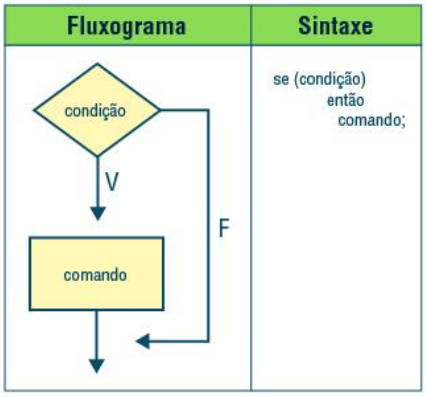
Isso só reforça a seguinte relação:

As estruturas algorítmicas de decisão só apresentam uma chance de execução do q (que pode ser um comando único denominado q ou um bloco com dois ou mais comandos chamados q), que é a **satisfação da condição p**, ou seja, o V(p) tem que ser Verdade.

03

2- Estrutura de Decisão Simples – com um comando

Observe a estrutura algorítmica abaixo.



Observe que esta estrutura algorítmica de decisão só apresenta uma chance de execução do comando q: é a satisfação da condição p, ou seja, o V(p) tem que ser Verdade.

Devido a esta característica, relacionando com lógica das proposições, considere a correspondência com a lógica das proposições:

```
se (e somente se) V(p)=V
    então
     execute q;
```

04

Um exemplo da lógica das proposições:

Se e somente se acordar cedo, irei à praia.

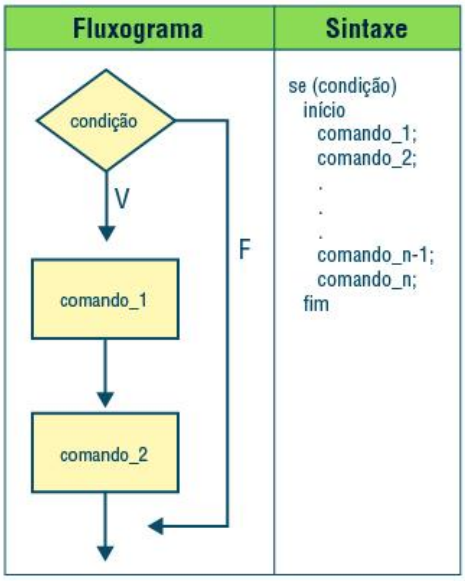
Analogamente, para esta estrutura algorítmica, tem-se:
Se e somente se a condição p for satisfeita, o programa executa o comando q.

Caso seja falsa a condição, o programa executa o primeiro comando localizado logo depois do fim da estrutura algorítmica, que, por sua vez, não faz parte dos comandos previstos pela estrutura em referência.

05

3 - Estrutura de Decisão Simples – Bloco de Comandos

Agora observe a estrutura algorítmica abaixo.



Nesta estrutura algorítmica de decisão só se apresenta uma chance de execução do conjunto de comandos de 1 até n: é a satisfação da condição p, ou seja, o V(p) tem que ser Verdade.

Vamos chamar o conjunto de comandos como **bloco de comandos q** e considere que a correspondência com a lógica das proposições:

```
se (e somente se) V(p)=V
 início
   execute q_1;
   execute q_2;
   ...
   execute q_n-1;
   execute q_n;
 fim
```

06

Um exemplo da lógica das proposições:

Se e somente se acordar cedo, irei à praia de manhã e irei ao cinema à tarde.

Analogamente, para esta estrutura algorítmica, tem-se:

Se e somente se a condição p for satisfeita, o programa executa as ações o bloco q.

Neste caso, perceba que o bloco de comandos utiliza a interpretação da conjunção, já que o programa deverá fazer a execução de todos os comandos de q_1 até q_n, ou de nenhum deles. Ou seja:

$q\_1 \wedge q\_2 \wedge \ldots \wedge q\_n$

A seguir, a linha 1 da tabela verdade de $(q\_1 \wedge q\_2 \wedge \ldots \wedge q\_n)$ representa a execução de todos os comandos do bloco, simbolizando o que deve acontecer quando a condição for Verdade.

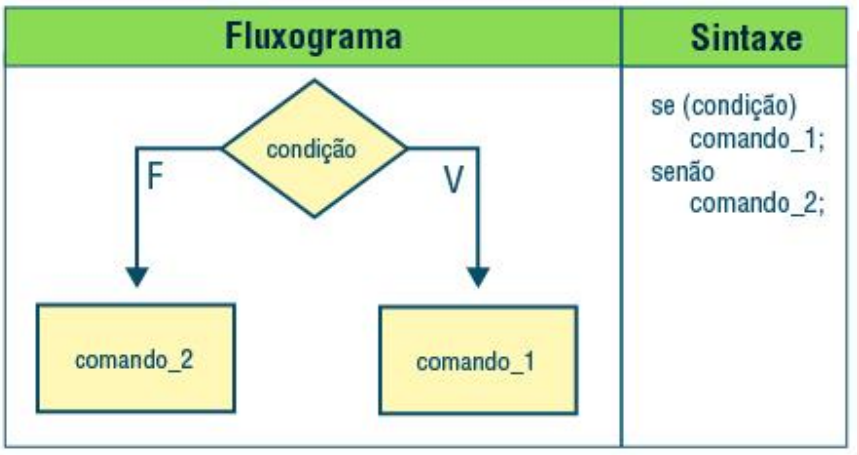
					A	B	X	Resultado final
	q_1	q_2	...	q_n	$q\_1 \wedge q\_2$	$A \wedge q\_3$	$\ldots \wedge q\_{n-1}$	$X \wedge q\_n$
1.	V	V	V	V	V	V	V	V
2.	V	V	V	F	F	F	F	F
3.	V	V	F	F	F	F	F	F
4.	V	F	F	F	F	F	F	F
5.	F	F	F	F	F	F	F	F
•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•
2^n.					F	F	F	F

Se a condição for falsa, lembre-se que o programa executará o primeiro comando que encontrar fora da estrutura algorítmica.

07

4 - Estrutura de Decisão Composta

Vejamos agora a seguinte estrutura de decisão composta.



Para esta estrutura de decisão, diferentemente das anteriores, dentro da própria estrutura está prevista uma ação a ser executada no caso da condição não ser satisfeita, devido à presença da linha *senão* e respectivo comando_2.

Devido a este aspecto, considere a correspondência com duas proposições:

```
se (e somente se) V(p)=V
  início
    execute q_1;
  fim
    se (e somente se) V(p)=F
  início
    execute q_2;
  fim
```

08

Um exemplo da lógica das proposições:

Se e somente se acordar cedo, irei à praia de manhã.

Se e somente se não acordar cedo, irei ao cinema à tarde.

Analogamente, para esta estrutura algorítmica, tem-se:

Se e somente se a condição p for satisfeita, o programa executa o comando q.

Se e somente se a condição p não for satisfeita, o programa executa o comando q_2.

Note que a execução do comando q exclui a possibilidade da execução de q_2, característica esta que é coincidente com a regra da disjunção exclusiva. Ou seja:

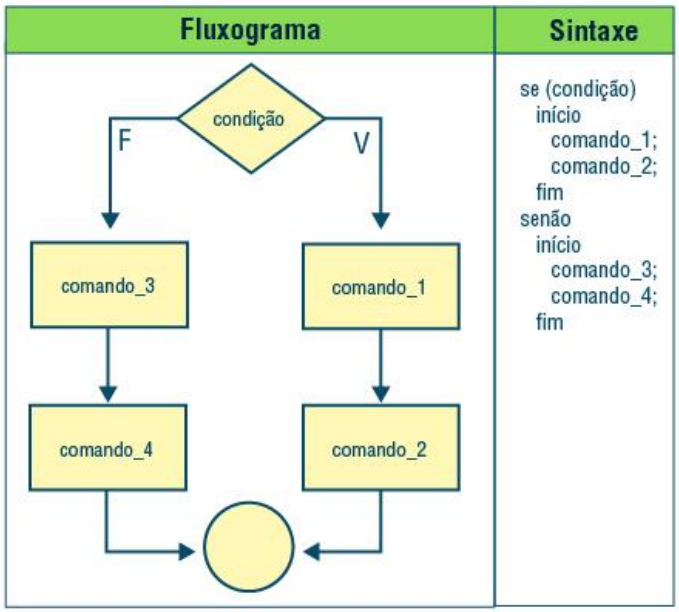
q V q_2

A tabela verdade de q V q_2 representa esta relação:

			Resultado final
	q	q_2	q ∨ q_2
1.	V	V	F
2.	V	F	V
3.	F	V	V
4.	F	F	F

09

4.1 - Estrutura de Decisão Composta – Bloco de Comandos



Dentro desta estrutura de decisão também já contempla a ação a ser executada no caso da condição não ser satisfeita, conforme evidencia a linha senão e respectivo bloco de comandos.

Devido a este aspecto, considere a correspondência com a lógica das proposições:

```
se (e somente se) V(p)=V
 início
  execute q_1;
  execute q_2;
 fim
se (e somente se) V(p)=F
 início
  execute q_3;
  execute q_4;
 fim
```

10

Um exemplo da lógica das proposições:

Se e somente se acordar cedo, irei à praia de manhã e irei almoçar fora.

Se e somente se não acordar cedo, irei ao cinema à tarde e ao supermercado à noite.

Analogamente, para esta estrutura algorítmica, tem-se:

Se e somente se a condição p for satisfeita, o programa executa o bloco de comandos q_1 e q_2.

Se e somente se a condição p não for satisfeita, o programa executa o bloco de comandos q_3 e q_4.

Note que a execução do bloco de comandos q_1 e q_2 exclui a possibilidade da execução do bloco de comandos q_3 e q_4, e vice-versa, característica esta que é coincidente com a regra da disjunção exclusiva. Além disso, é importante lembrar que o comportamento de execução dos blocos de comandos corresponde à lógica da conjunção. Ou seja:

$$(q\_1 \wedge q\_2) \vee (q\_3 \wedge q\_4)$$

A tabela verdade abaixo mostra esta relação:

			Resultado final
	A	B	$A \veebar B$
1.	V	V	**F**
2.	V	F	**V**
3.	F	V	**V**
4.	F	F	**F**

Sendo,
A: q_1 ∧ q_2
B: q_3 ∧ q_4

11

5 - Comando Escolha

Sintaxe

```
Escolha (opção)
  Início
  Caso 1:
        comando_1;
        comando_2;
  Caso 2:
        comando_3;
        comando_4;
  Caso 3:
        comando_1;
        comando_2;
  fim
```

Nesta estrutura de decisão temos blocos de comandos, cada um com uma condição exclusiva. O bloco de comando será executado se e somente se a respectiva condição for satisfeita.
Considere então a seguinte correspondência com a lógica das proposições:

```
se (e somente se) opção=1
        execute q_1;
        execute q_2;
se (e somente se) opção=2
        execute q_3;
        execute q_4;
se (e somente se) opção=3
        execute q_5;
        execute q_6;
```

Caso a escolha feita pelo usuário do programa não esteja dentre as opções disponíveis na estrutura de decisão, então não haverá condição satisfeita e não haverá execução de qualquer dos blocos de comandos nela previsto. Neste caso, o computador executa a próxima ação localizada fora da estrutura algorítmica.

Um exemplo da lógica das proposições:

Se e somente se acordar cedo, irei à praia de manhã e irei almoçar fora.

Se e somente se não chover, irei ao cinema à tarde e ao supermercado à noite.

Se e somente se conseguir uma carona, pegarei meu carro na oficina e farei uma visita ao Lucas.

Analogamente, para esta estrutura algorítmica, tem-se:

Se e somente se for satisfeita a condição da opção ser 1, o programa executa o bloco de comandos q_1 e q_2.

Se e somente se for satisfeita a condição da opção ser 2, o programa executa o o bloco de comandos q_3 e q_4.

Se e somente se for satisfeita a condição da opção ser 3, o programa executa o bloco de comandos q_5 e q_6.

Note que a execução de qualquer um dos blocos de comandos exclui a possibilidade de execução de qualquer um dos outros blocos de comandos, característica esta que é coincidente com a regra da disjunção exclusiva. Além disso, já vimos que o comportamento da execução dos blocos de comandos corresponde à lógica da conjunção. Ou seja:

$$(q\_1 \wedge q\_2) \vee (q\_3 \wedge q\_4) \vee (q\_5 \wedge q\_6)$$

A tabela verdade abaixo mostra esta relação:

			X	Resultado final
A	B	C	$A \veebar B$	$X \veebar C$
V	V	V	F	V
V	V	F	F	F
V	F	V	V	F
V	F	F	V	V
F	V	V	V	F
F	V	F	V	V
F	F	V	F	V
F	F	F	F	F

Sendo,

A: $q\_1 \wedge q\_2$

B: $q\_3 \wedge q\_4$

C: $q\_5 \wedge q\_6$

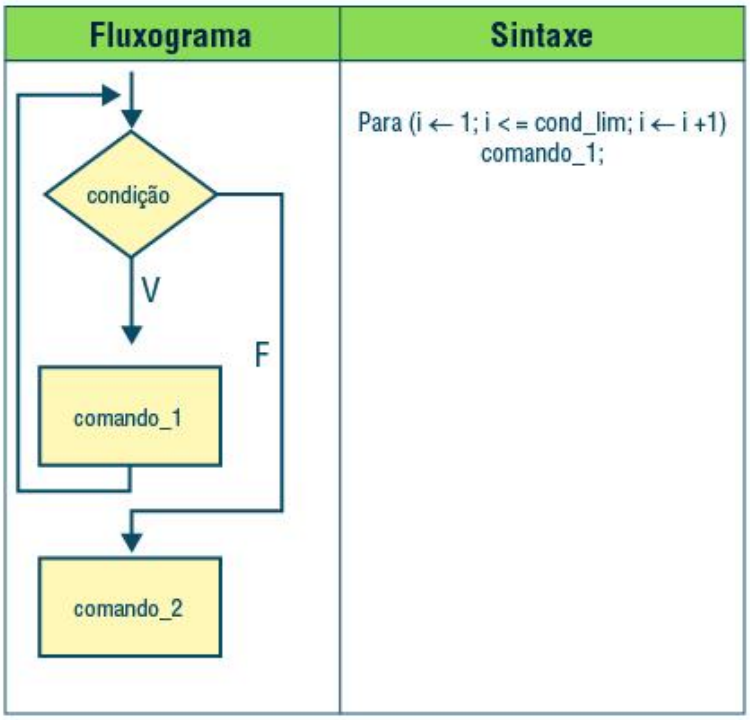
13

6 - Estruturas de repetição e a lógica das proposições

A lógica do raciocínio baseado na veracidade de uma condição é mantida nas estruturas algorítmicas de repetição, mas as referidas estruturas acrescentam a capacidade do programa **repetir** a mesma tarefa um número desejado de vezes.

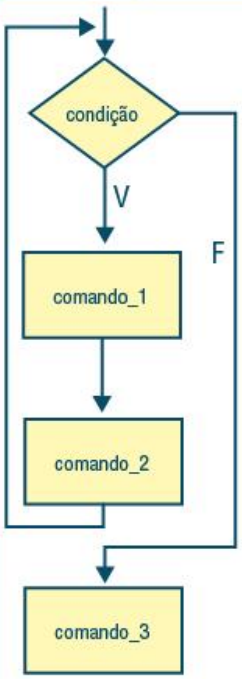
Dessa forma, as estruturas de repetição são também compostas por um ou mais comandos, cuja execução depende da satisfação de uma condição. No entanto, o controle quanto à satisfação da condição pode ser tanto pela análise simples da veracidade, como pelo estabelecimento de um intervalo que limite o quanto a repetição será executada.

- **Estrutura Para**



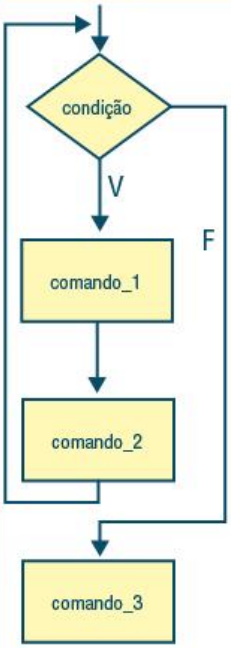
14

- **Estrutura Para – Bloco de Comandos**

Fluxograma	Sintaxe
	Para (i ← 1; i < = cond_lim; i ← i +1) comando_1; início comando_1; comando_2; fim

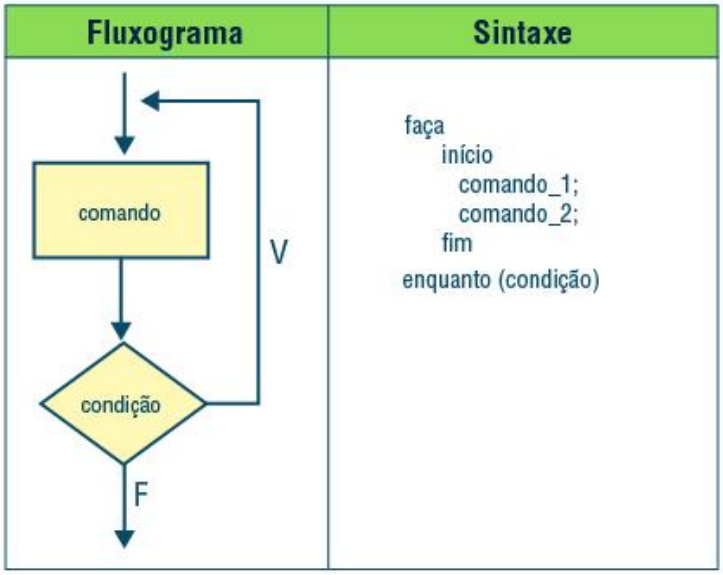
15

- **Estrutura Enquanto**

Fluxograma	Sintaxe
	enquanto (condição) início comando_1; comando_2; fim

16

- **Estrutura Faça-Enquanto**



Note que a lógica do raciocínio é realmente similar ao das estruturas de decisão, sendo que com a funcionalidade de repetição. Assim, permanecem todas as correspondências que vimos nas estruturas de decisão quanto à lógica das proposições, pois continuam válidos os aspectos relacionados à capacidade de decidir entre duas ou mais possibilidades de execução.

O aspecto acrescido, que é a capacidade de **repetir** um ou mais comandos até o limite desejado, traz o diferencial à relação entre estes tipos de estruturas algorítmicas e a lógica das proposições:

A checagem quanto à satisfação da condição é feita mais de uma vez durante a utilização do programa. Assim, garante-se que a execução prevista será repetida até o momento estabelecido pela estrutura algorítmica.

A cada checagem há a possibilidade de mudança de valor lógico da condição, e, consequentemente, de alteração na interpretação e no curso da execução. No entanto, nas estruturas de decisão, esta checagem ocorre uma única vez, definindo por definitivo a interpretação e respectiva execução.

17

RESUMO

Neste módulo foram abordadas as convergências de interpretações entre as regras das operações lógicas e as regras das estruturas algorítmicas, conforme detalhamento dos dois grandes grupos: estruturas algorítmicas de decisão e estruturas algorítmicas de repetição.

A lógica do raciocínio das operações condicional e bicondicional é correspondente à utilizada pelas estruturas algorítmicas de decisão, que atribuem ao programa a capacidade de decidir entre duas ou mais possibilidades de execução.

As estruturas algorítmicas de decisão só apresentam uma chance de execução do q (que pode ser um comando único denominado q ou um bloco com dois ou mais comando chamado q), que é a satisfação da condição p, ou seja, o V(p) tem que ser Verdade.

As estruturas de repetição são também compostas por um ou mais comandos, cuja execução depende da satisfação de uma condição. No entanto, o controle quanto à satisfação da condição pode ser tanto pela análise simples da veracidade, como pelo estabelecimento de um intervalo que limite o quanto a repetição será executada.