1

Lógica Formal

Objetivos do Capítulo

Após estudar este capítulo, o leitor deve ser capaz de:

- Reconhecer e trabalhar com os símbolos formais que são usados nas lógicas proposicional e de predicados
- Achar o valor-verdade de uma expressão na lógica proposicional
- Achar o valor-verdade de alguma interpretação de uma expressão na lógica de predicados
- Usar a lógica de predicados para representar sentenças da língua portuguesa
- Construir demonstrações formais nas lógicas proposicional e de predicados, e usá-las para determinar a validade de um argumento da língua portuguesa
- Entender como a linguagem de programação Prolog é constituída em função da lógica de predicados
- Provar matematicamente a correção de programas que usam comandos de atribuição e comandos condicionais

A exemplo de qualquer outra ciência, a Ciência da Computação depende da Matemática para obter um vocabulário preciso, uma notação poderosa, abstrações úteis e um raciocínio rigoroso. O objetivo deste livro é melhorar nosso entendimento da linguagem, das ferramentas e dos processos de raciocínio da Matemática que são usados na Ciência da Computação.

Este capítulo introduz a Lógica Formal, que delineia o método organizado e cuidadoso de pensar que caracteriza qualquer investigação científica ou qualquer outra atividade de raciocínio. Além disso, a lógica formal tem aplicações diretas na Ciência da Computação. A última seção deste capítulo explora uma linguagem baseada na lógica e no uso da Lógica Formal objetivando verificar a correção de programas de computadores. Ainda, a lógica de circuitos (a lógica que rege os circuitos de computadores) é um análogo direto da lógica de sentenças deste capítulo. Estudaremos este tipo de lógica no Cap. 7.

Seção 1.1 Sentenças, Representação Simbólica e Tautologias

Geralmente nos expressamos, em português, através de interrogações e exclamações, mas, para comunicar fatos ou informações, usamos sentenças. Tecnicamente, uma **sentença** (ou **proposição)** é uma frase que pode ser apenas verdadeira ou falsa.

EXEMPLO 1 Considere o seguinte:

a. Dez é menor do que sete.
b. Como vai você?
c. Ela é muito talentosa.
d. Existem formas de vida em outros planetas do universo.

A frase (a) é uma sentença porque é falsa. Como o item (b) é uma pergunta, não pode ser considerado nem verdadeiro nem falso. Não tem valor-verdade e, portanto, não é uma sentença. Na frase (c) a palavra *ela* é uma variável e a frase não é verdadeira nem falsa, pois *ela* não está especificada; portanto, (c) não é uma sentença. A frase (d) é uma sentença porque é verdadeira ou falsa; independentemente de sermos capazes de decidir qual dos dois.

Conectivos e Valores-Verdade

Para enriquecermos nossas conversas não nos limitamos ao uso de simples sentenças. Ao contrário, as combinamos com o uso de conectivos a fim de criarmos sentenças compostas, cujo valor-verdade depende dos valores-verdade de cada sentença que o compõe e dos conectivos usados. Um conectivo comum é a palavra *e*. (Palavras como *mas* e *também* expressam diferentes significados, mas têm o mesmo efeito sobre os valores-verdade.) Se combinarmos as duas sentenças verdadeiras "Elefantes são grandes" e "Bolas são redondas" devemos considerar a sentença resultante "Elefantes são grandes e bolas são redondas" como verdadeira. Na lógica, usamos o símbolo A para denotar o conectivo lógico *e* e letras maiúsculas para denotar as sentenças (ao que chamaremos de símbolos proposicionais). Valores-verdade são atribuídos aos símbolos proposicionais. Concordamos, então, que se A e B são verdadeiras, *A* A *B* (leia-se *"A* e B") deve ser considerada verdadeira.

PRATICA 1
a. Se *A* é verdadeira e *B* é falsa, que valor você atribuiria a *A* ^ *B?*
b. Se *A* é falsa e *B* é verdadeira, que valor você atribuiria a *A* ^ *B?*
c. Se *A* e *B* são ambas falsas, que valor você atribuiria a *A* ^ *B?*

(As respostas aos exercícios práticos estão no fim do livro.) •

A expressão *A* A *B* é chamada a **conjunção** de *A* e *B;* e *A* e *B* são chamados os **fatores** da expressão. Podemos resumir os efeitos das conjunções através do uso da **tabela-verdade** apresentada na Tabela 1.1. Em cada linha da tabela-verdade os valores-verdade são atribuídos aos símbolos proposicionais e o valor-verdade resultante da composição da expressão é, então, mostrado.

A	*B*	*A^B*
V	V	V
V	F	F
F	V	F
F	F	F

Tabela 1.1

A	*B*	*A^B*
V	V	V
V	F	
F	V	
F	F	

Outro conectivo é a palavra *ou*, denotada pelo símbolo $\vee$. A expressão $A \vee B$ (leia-se "A ou B") é chamada **disjunção** de A e B e Ae B são chamados de **parcelas** da expressão. *Se A e B* forem ambos verdadeiros, $A \vee B$ deverá ser considerada verdadeira, nos dando a primeira linha da tabela-verdade para a disjunção (veja a Tabela **1.2).**

PRATICA 2 Use o que entendeu da palavra *ou* para completar a tabela-verdade para a disjunção, isto é, a Tabela 1.2 •

As sentenças podem ainda ser combinadas na forma "se sentença 1, então sentença 2". Se *A* denota a sentença 1 e *B* denota a sentença 2, a sentença composta deve ser denotada por $A \rightarrow B$ (leia-se "A implica *B").* O conectivo lógico aqui é a **implicação** e indica que a verdade de *A* implica ou leva à verdade de *B.* Existem outras maneiras de expressar $A \rightarrow B$ na linguagem quotidiana, tal como "A é condição suficiente para B", "A somente se *B", "B* é conseqüência de A". Na expressão $A \rightarrow B$, *A* constitui a sentença **antecedente** e *B* a sentença **conseqüente.**

EXEMPLO 2 A sentença "Fogo é uma condição necessária para fumaça" pode ser reformulada como "Se há fumaça, então há fogo". O antecedente é "há fumaça", e o conseqüente é "há fogo". •

PRATICA 3 Indique o antecedente e o conseqüente em cada uma das seguintes sentenças. *(Dica:* reescreva as frases na forma se-então.) •

a. Se a chuva continuar, o rio vai transbordar.
b. Uma condição suficiente para a falha de uma rede é que a chave geral pare de funcionar.
c. Os abacates só estão maduros quando estão escuros e macios.
d. Uma boa dieta é uma condição necessária para um gato saudável. •

A tabela-verdade para a implicação é menos óbvia do que para a conjunção e disjunção. Para compreendermos sua definição, vamos supor que seu colega de quarto diga "Se eu me formar nesta primavera, vou tirar férias na Flórida." Se ele, de fato, se formar na primavera e tirar suas férias na Flórida, a sentença foi verdadeira. Se A e *B* forem ambas verdadeiras, consideraremos a implicação $A \rightarrow B$ verdadeira. Se o seu colega se formar e não tirar as férias na Flórida, seu comentário consistiu em uma sentença falsa. Quando A é verdadeira e *B* é falsa, consideramos $A \rightarrow B$ falsa. Suponhamos agora que seu colega não se formou. Independentemente de ele tirar ou não férias na Flórida, não poderemos acusá-lo de ter formulado uma sentença falsa e lhe daremos o benefício da dúvida. Por convenção, aceitamos $A \rightarrow B$ como verdadeira se A for falsa, independentemente do valor-verdade de *B.*

PRATICA 4 Resuma a explanação acima, escrevendo a tabela-verdade de $A \rightarrow B$. •

A	*B*	$A \rightarrow B$	$B \rightarrow A$	$(A \rightarrow B) \wedge (B \rightarrow A)$
V	V	V	V	V
V	F	F	V	F
F	V	V	F	F
F	F	V	V	V

Tabela 1.3

Suponhamos que $A \rightarrow B$ seja verdadeira. Então, de acordo com a tabela-verdade da implicação, *B* pode ser verdadeira ou A pode ser falsa. Portanto, a despeito do fato de A ser verdadeira implicar que *B* também o seja, *B* ser verdadeira não implica que A o seja. A frase "S é uma condição suficiente para A" para denotar $A \rightarrow B$ apenas significa que, se A for verdadeira, *B* também o será.

O conectivo de **equivalência** é denotado pelo símbolo $\leftrightarrow$. A expressão $A \leftrightarrow B$ é a abreviação de $(A \rightarrow B)$ A $(B \rightarrow A)$. Podemos escrever a tabela-verdade para a equivalência construindo, parte por parte, a tabela para $(A \rightarrow B)$ A $(B \rightarrow A)$, como na Tabela 1.3. Perceba, nesta tabela, que $A \leftrightarrow B$ é verdadeira exatamente quando A e *B* têm o mesmo valor-verdade. A expressão $A \leftrightarrow B$ *é* normalmente lida como "A se, e somente se. B".

Os conectivos que vimos até agora são chamados de **conectivos binários** pois eles unem duas expressões a fim de produzir uma terceira. Vamos agora considerar um **conectivo unário,** isto é, um conectivo que atua em uma única expressão para produzir uma outra. A **negação** é um conectivo unário. A negação de A, A ' é lida como "não A", "A é falsa" ou "A não é verdade". Isto não quer dizer que A' sempre tenha um valor-verdade falso, mas que o valor-verdade de A' é o contrário do de A. (Alguns textos denotam A ' como A.)

PRATICA 5 Escreva a tabela-verdade para A'. (Serão necessárias apenas duas linhas.) •

EXEMPLO 3 Se A é a sentença "Vai chover amanhã", a sentença A' é "Não é verdade que vai chover amanhã", que pode ser reescrita como "Não vai chover amanhã". •

Achar a negativa de uma sentença composta pode exigir algum esforço. Se P for a sentença "Peter é alto e magro", então a sentença P' será "É falso que Peter seja alto e magro", que pode ser reformulada como "Peter não é alto ou não é magro". Perceba que esta sentença *não* é a mesma que "Peter é baixo e gordo". Se P for a sentença "O rio é raso ou poluído", então P' é a sentença "É falso que o rio seja raso ou poluído", que pode ser reescrita como "O rio nem é raso nem é poluído" ou ainda "O rio é profundo e despoluído". No entanto, P' *não* é equivalente a "O rio não é raso ou não é poluído".

PRATICA 6 Qual das frases a seguir representa A' se A é a sentença "Julie adora manteiga mas detesta nata"?

a. Julie detesta manteiga e nata.
b. Julie não gosta de manteiga ou nata.
c. Julie não gosta de manteiga mas adora nata.
d. Julie detesta manteiga ou adora nata. •

Podemos encadear sentenças, seus conectivos e os parênteses (ou colchetes) para obtermos novas expressões, tal como em

$$(A \rightarrow B) \wedge (B \rightarrow A)$$

Naturalmente, como nas linguagens de programação, regem algumas *regras de sintaxe* (regras sob as quais as cadeias são válidas); por exemplo,

$$A)) \wedge \wedge \rightarrow BC$$

não deve ser considerada uma cadeia válida. Expressões que formam cadeias válidas são chamadas de **fórmulas bem-formuladas** ou **wffs** (de *well-formed formulas).* A fim de reduzir o número de parênteses necessários em uma wff, estipulamos uma ordem na qual os conectivos são aplicados. Esta "ordem de precedência" é:

1. Conectivos dentro de parênteses, dos mais internos para os mais externos
2. '
3. $\wedge$, $\vee$
4. $\rightarrow$
5. $\leftrightarrow$

Isto significa que a expressão $A \vee B'$ significa $A \vee (B')$ e não $(A \vee B)'$. Analogamente, $A \vee B \rightarrow C$ significa $(A \vee B) \rightarrow C$ e não $A \vee (B \rightarrow C)$.

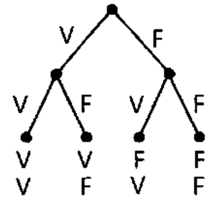
Wffs compostas de letras representativas de sentenças e conectivos têm valores-verdade que dependem dos valores-verdade atribuídos aos símbolos proposicionais. Escrevemos as tabelas-verdade para qualquer wff montando as partes que as compõem, como fizemos para $(A \rightarrow B) \wedge (B \rightarrow A)$.

EXEMPLO 4 A tabela-verdade para a wff $A \vee B' \rightarrow (A \vee B)'$ é dada na Tabela **1**.4. •

A	B	B'	$A \vee B'$	$A \vee B$	$(A \vee B)'$	$A \vee B' \rightarrow (A \vee B)'$
V	V	F	V	V	F	F
V	F	V	V	V	F	F
F	V	F	F	V	F	V
F	F	V	V	F	V	V

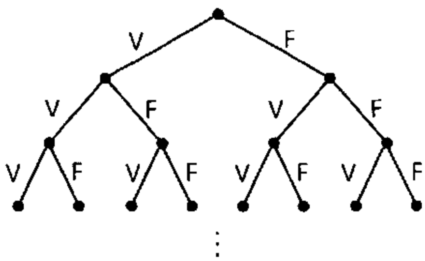
Tabela 1.4

Se estivermos montando uma tabela-verdade para uma wff que contenha n símbolos proposicionais diferentes, quantas linhas terá a tabela? Das tabelas feitas até agora, sabemos que uma wff com apenas um símbolo proposicional tem duas linhas em sua tabela-verdade e uma wff com dois símbolos proposicionais tem quatro linhas. O número de linhas é igual ao número de combinações verdadeiro-falso possíveis entre as letras



Letras de afirmações	Possibilidades
1	$2 = 2^1$ nós
2	$4 = 2^2$ nós

(a)



Letras de afirmações	Possibilidades
1	$2 = 2^1$ nós
2	$4 = 2^2$ nós
3	$8 = 2^3$ nós
n	(2^n nós)

(b)

Figura 1.1

das combinações. O primeiro símbolo proposicional tem dois valores possíveis, V e F. Para cada um desses valores, o segundo símbolo proposicional também tem dois valores possíveis. A Fig. 1.1a ilustra isto na forma de uma "árvore" de dois níveis com quatro folhas, que mostram as quatro combinações possíveis de V e F para dois símbolos proposicionais. Para n símbolos proposicionais estendemos a árvore a n níveis, como na Fig. 1.1 .b. O número total de folhas será igual, portanto, a 2^n O número total de linhas em uma tabela-verdade para n símbolos proposicionais será também 2".

Essa estrutura de árvore nos mostra como enumerar todas as combinações V-F dentre os n símbolos proposicionais quando precisamos montar uma tabela-verdade. Se lermos cada nível da árvore de baixo para cima, veremos que os valores V-F para o símbolo proposicional n (a que compõe a última coluna da tabela-verdade) desdobra cada valor da linha n — 1 em dois valores e os da linha n — *2* em quatro valores e assim por diante. Portanto, uma tabela-verdade para três símbolos proposicionais começaria como mostrado na Tabela 1.5. Os valores para a sentença C variam, para os valores da sentença B em grupos de dois e para os valores da sentença A em grupos de quatro, resultando em uma versão horizontal de uma árvore. (Lendo as linhas de cima para baixo e usando 1 para os V e 0 para os F veremos que estamos apenas contando a partir de zero em binário.)

A	*B*	*C*
V	V	V
V	V	F
V	F	V
V	F	F
F	V	V
F	V	F
F	F	V
F	F	F

Tabela 1.5

PRATICA 7 Construa as tabelas-verdade para as seguintes wffs.

a. $(A \to B) \leftrightarrow (B \to A)$ (lembre-se que $C \leftrightarrow$ D só é verdadeira quando C *e* D têm o mesmo valor- verdade)

b. $(A \vee A') \to (B \wedge B')$

c. $[(A \wedge B') \to C']'$

d. $(A \to B) \leftrightarrow (B' \to A')$ •

Tautologias

Uma wff como a (d) da Prática 7, cujos valores-verdade são sempre verdadeiros, é chamada uma **tautologia.** Uma wff como a do item (b), cujos valores-verdade são sempre falsos, é chamada uma **contradição.** Quando uma wff composta na forma $P \leftrightarrow Q$ *é* uma tautologia, como na Prática 7(d), os valores-verdade de P e Q conferem para todas as colunas da tabela-verdade. Neste caso, P e Q são chamadas de **wffs equivalentes** e denotadas por $P \Leftrightarrow Q$. Desta forma, $P \Leftrightarrow Q$ indica um fato, a saber: a wff particular $P \leftrightarrow Q$ *é* uma tautologia.

Vamos listar algumas equivalências básicas, provar uma ou duas delas através da construção das tabelas-verdade e deixar as demais como exercícios. Representaremos qualquer contradiçãcrpor 0 e qualquer tautologia por 1.

Algumas Equivalências Tautológicas

1a. $A \vee B \Leftrightarrow B \vee A$	**1b.** $A \wedge B \Leftrightarrow B \wedge A$	(propriedades comutativas)
2a. $(A \vee B) \vee C \Leftrightarrow A \vee (B \vee C)$	**2b.** $(A \wedge B) \wedge C \Leftrightarrow A \wedge (B \wedge C)$	(propriedades associativas)
3a. $A \vee (B \wedge C) \Leftrightarrow (A \vee B) \wedge (A \vee C)$	**3b.** $A \wedge (B \vee C) \Leftrightarrow (A \wedge B) \vee (A \wedge C)$	(propriedades distributivas)
4a. $A \vee 0 \Leftrightarrow A$	**4b.** $A \wedge 1 \Leftrightarrow A$	(propriedades de identidade)
5a. $A \vee A' \Leftrightarrow 1$	**5b.** $A \wedge A' \Leftrightarrow 0$	(propriedades complementativas)

(Perceba que 2a nos leva a escrever $A \vee B \vee C$ sem a necessidade de parênteses; analogamente, 2b nos leva a escrever $A \wedge B \wedge C$.)

EXEMPLO 5 A tabela-verdade da Tabela 1.6 verifica a equivalência 1 a, a propriedade comutativa da disjunção, e a da Tabela 1.7 verifica a 4b, a propriedade da identidade para a conjunção. Perceba que foram necessárias apenas duas linhas para a Tabela 1.7 porque 1 (uma tautologia) não pode assumir valores falsos. •

A	B	$A \vee B$	$B \vee A$	$A \vee B \leftrightarrow$5 v A
V	V	V	V	V
V	F	V	V	V
F	V	V	V	V
F	F	F	F	V

Tabela 1.6

A	1	$A \wedge 1$	$A \wedge 1 \leftrightarrow A$
V	V	V	V
F	V	F	V

Tabela 1.7

PRATICA 8 Verifique a equivalência 5a. •

As equivalências em nossa lista são grupadas em cinco pares. Em cada par, uma equivalência pode ser obtida da outra substituindo-se os $\wedge$ por $\vee$, $\vee$ por $\wedge$, 0 por 1 e 1 por 0. Cada equivalência em um par é chamada a **dual** da outra. Esta lista de equivalência aparece de uma forma mais geral no Cap. 7.

Duas outras equivalências muito úteis são as **leis de De Morgan,** assim chamadas devido ao matemático inglês do século XIX Augustus De Morgan, que primeiro as formulou. Essas leis são:

$$(A \vee B)' \Leftrightarrow A' \wedge B' \quad \text{e} \quad (A \wedge B)' \Leftrightarrow A' \vee B'$$

Cada qual é a dual da outra. As leis de De Morgan nos ajudam a representar a negação de sentenças compostas, como na Prática 6.

Suponha que P e Q são equivalentes e que P aparece como uma componente de uma wff R grande. O que acontece se substituirmos Q por P? Por mais que os valores-verdade mudem, não haverá qualquer alteração. Denotemos por R_Q a wff obtida pela substituição de P por Q em R. Se construirmos as tabelas-verdade para R e para R_Q, a cada linha os valores de Q e P coincidem; logo, a cada linha, os valores de R e R_Q também coincidem. Portanto. R_Q *é* equivalente à wff original R.

EXEMPLO 6 Seja R $(A \to B) \to B$ e seja P $A \to B$. Da Prática 7(d), sabemos que Pé equivalente a $Q = B' \to A'$. Substituindo P por Q. obtemos $R_Q = (B' \to A') \to B$. As tabelas-verdade para R e R_Q podem ser vistas nas Tabelas 1.8 e 1.9. Os valores-verdade de $A \to B$ e $B' \to A'$ coincidem para todas as linhas, portanto os valores-verdade de R e R_Q coincidem para todas as linhas. Logo R e R_Q são equivalentes. •

A	B	$A \rightarrow B$	$(A \rightarrow B) \rightarrow B$
V	V	V	V
V	F	F	V
F	V	V	V
F	F	V	F

Tabela 1.8

A	B	A'	B'	$B' \rightarrow A'$	$(B' \rightarrow A') \rightarrow B$
V	V	F	F	V	V
V	F	F	V	F	V
F	V	V	F	V	V
F	F	V	V	V	F

Tabela 1.9

Equivalências Tautológicas

Propriedades Comutativas:	$A \vee B \Leftrightarrow B \vee A$	$A \wedge B \Leftrightarrow B \wedge A$
Propriedades Associativas:	$(A \vee B) \vee C \Leftrightarrow$ $A \vee (B \vee C)$	$(A \wedge B) \wedge C \Leftrightarrow$ $A \wedge (B \wedge C)$
Propriedades Distributivas:	$A \vee (B \wedge C) \Leftrightarrow$ $(A \vee B) \wedge (A \vee C)$	$A \wedge (B \vee C) \Leftrightarrow$ $(A \wedge B) \vee (A \wedge C)$
Propriedades de Identidade:	$A \vee 0 \Leftrightarrow A$	$A \wedge 1 \Leftrightarrow A$
Propriedades Complementativas:	$A \vee A' \Leftrightarrow 1$	$A \wedge A' \Leftrightarrow 0$
Leis de De Morgan:	$(A \vee B)' \Leftrightarrow A' \wedge B'$	$(A \wedge B)' \Leftrightarrow A' \vee B'$
Propriedades Idempotentes:	$A \vee A \Leftrightarrow A$	$A \wedge A \Leftrightarrow A$
Dupla Negativa:	$(A')' \Leftrightarrow A$	
Reescrevendo a Implicação:	$(A \rightarrow B) \Leftrightarrow A' \vee B$	
Contraposição:	$(A \rightarrow B) \Leftrightarrow (B' \rightarrow A')$	
Prova Condicional	$A \rightarrow (B \rightarrow C) \Leftrightarrow$ $(A \wedge B) \rightarrow C$	

Apesar de existirem infinitas tautologias, o quadro anterior resume as equivalências mais úteis (que expressam certas tautologias), incluindo aquelas que já havíamos listado anteriormente. Essas equivalências podem ser usadas, como vimos, para reescrever wffs. Como antes, 0 é qualquer contradição e 1 é qualquer tautologia.

Conectivos Lógicos e Programação

Os conectivos lógicos E, OU e NÃO (ou, mais comumente seus equivalentes em inglês AND, OR e NOT) são oferecidos pela maioria das linguagens de programação. Esses conectivos, de acordo com as tabelas-verdade que definimos, agem sobre combinações de expressões verdadeiras e falsas a fim de produzir um valor-verdade final. Desses valores provém a capacidade de tomada de decisão fundamental ao controle do fluxo de programas de computadores. Desta forma, em um desvio condicional de um programa, se o valor-verdade de uma determinada expressão for verdadeiro, o programa irá executar um trecho de seu código; se o valor for falso, o programa executa, em seguida, outro trecho de seu código. Se a expressão condicional for substituída por uma expressão mais simples equivalente, o valor-verdade da expressão e, portanto, o controle do fluxo do programa não serão afetados, mas o novo código torna-se mais simples de ser entendido e poderá ser executado mais rapidamente.

EXEMPLO 7 Vejamos o seguinte comando na linguagem de programação Pascal:

```
if (fluxoext > fluxoint)
      and not ((fluxoext > fluxoint) and (pressão < 1000)) then
            UmProcedimento(lista de parâmetros)
else
            OutroProcedimento(lista de parâmetros);
```

A expressão condicional aqui tem a seguinte forma

$$A \wedge (A \wedge B)'$$

onde A é *fluxoext > fluxoint* e B é *pressão* < 1000. Esta expressão pode ser simplificada substituindo-se algumas subexpressões por suas expressões equivalentes.