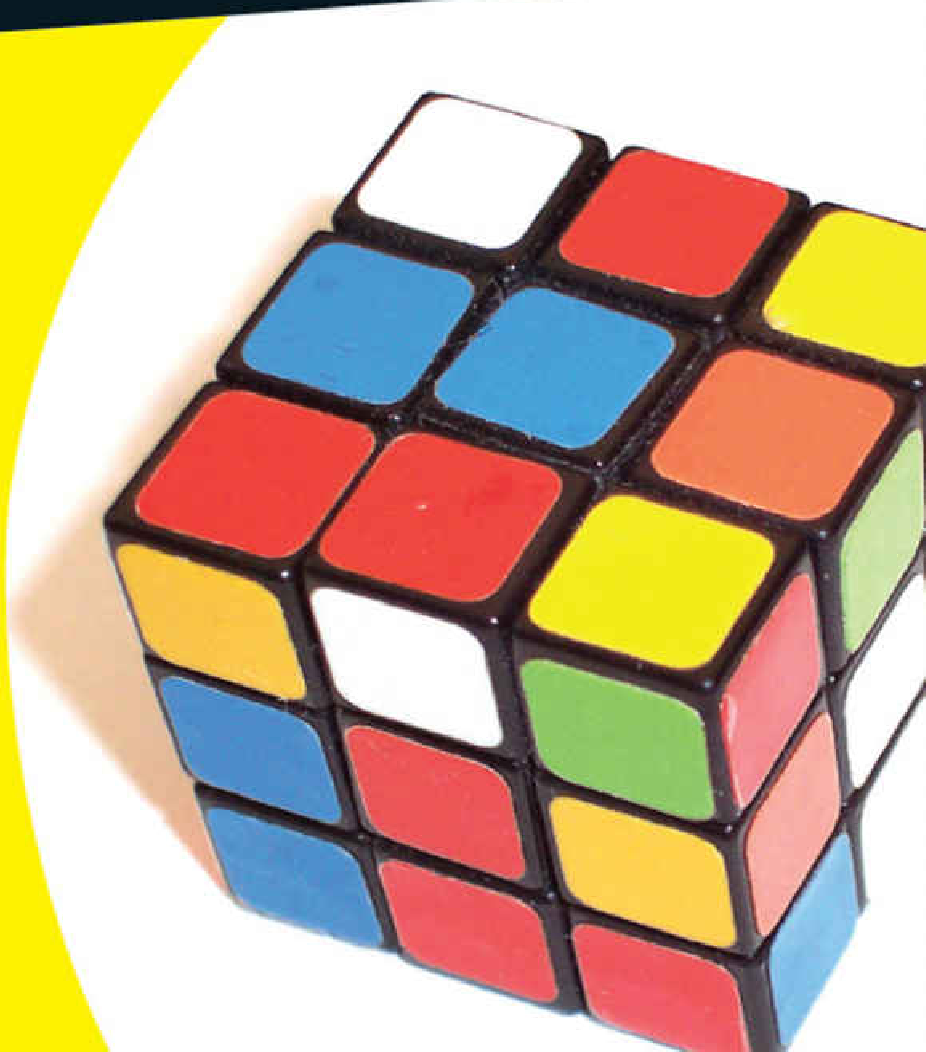
A forma divertida e fácil de aprender argumentos lógicos, deduções e provas

Lógica

PARA LEIGOS®

FOR DUMMIES®



Aprenda:

- A simplificar conclusões lógicas
- Tabelas Verdade e Tabelas Rápidas
- Regras de equivalência
- Grandes nomes da Lógica

Tornando tudo mais fácil!

Mark Zegarelli

Especialista e criador de quebra-cabeças lógicos

Lógica Para Leigos

Folha de Cola

Operadores de Lógica Sentencial

Operador	Nome Técnico	Significado	Exemplo
$\sim$	Negação	Não	$\sim x$
$\wedge$	Conjunção	E	$x \wedge y$
$\vee$	Disjunção	Ou	$x \vee y$
$\rightarrow$	Condicional	Se...então	$x \rightarrow y$
$\leftrightarrow$	Bicondicional	Se e somente se	$x \leftrightarrow y$

Tabelas Verdade

X	*~X*
V	*F*
F	*V*

x	y	$x \wedge y$	$x \vee y$	$x \rightarrow y$	$x \leftrightarrow y$
V	V	V	V	V	V
V	F	F	V	F	F
F	V	F	V	V	F
F	F	F	F	V	V

Dicas para Fazer um Exame de Lógica

- Comece com uma rápida olhada por toda a prova para perceber do que se trata.
- Comece por um problema mais fácil.
- Preencha as tabelas verdade, coluna por coluna.
- Se você sabe que cometeu um erro, diga – você pode receber alguns pontos pela questão.
- Se o tempo for curto, acabe as coisas mais entediantes.
- Verifique – e verifique novamente – o seu trabalho.

Regras de Implicação

Modus Ponens (MP)

$x \rightarrow y, x : y$

Modus Tollens (MT)

$x \rightarrow y, \sim y : \sim x$

Conjunção (Conj.)

$x, y : x \wedge y$

Simplificação (Simp.)

$x \wedge y : x$

$x \wedge y : y$

Adição (Ad.)

$x : x \vee y$

Silogismo Disjuntivo (SD)

$x \vee y, \sim x : y$

$x \vee y, \sim y : x$

Silogismo Hipotético (SH)

$x \rightarrow y, y \rightarrow z : x \rightarrow z$

Dilema Construtivo (DC)

$w \vee x, w \rightarrow y, x \rightarrow z : y \vee z$

Para Leigos: A série de livros para iniciantes que mais vende no mundo.

Lógica Para Leigos

Folha de Cola

Identidade e Regras de Quantitativa

Regra de Identidade (ID): $\forall x,\ x = y : \forall y$

Identidade Reflexiva (IR): $\forall x\ [x = x]$

Regras de Equivalência para a Lógica Sentencial

Dupla Negação (DN)

$x :: \sim(\sim x)$

Contraposição (Contra)

$x \to y :: \sim y \to \sim x$

Implicação (Impl)

$x \to y :: \sim x \vee y$

Exportação (Exp)

$x \to (y \to z) :: (x \wedge y) \to z$

Comutação (Comm)

$x \wedge y :: y \wedge x$

$x \vee y :: y \vee x$

Associação (Assoc)

$(x \wedge y) \wedge z :: x \wedge (y \wedge z)$

$(x \vee y) \vee z :: x \vee (y \vee z)$

Distribuição (Dist)

$x \wedge (y \vee z) :: (x \wedge y) \vee (x \wedge z)$

$x \vee (y \wedge z) :: (x \vee y) \wedge (x \vee z)$

Teorema de DeMorgan (DeM)

$\sim(x \wedge y) :: \sim x \vee \sim y$

$\sim(x \vee y) :: \sim x \wedge \sim y$

Tautologia (Taut)

$x \wedge x :: x$

$x \vee x :: x$

Equivalência (Equiv)

$x \leftrightarrow y :: (x \to y) \wedge (y \to x)$

$x \leftrightarrow y :: (x \wedge y) \vee (-x \wedge -y)$

Quantificação para Lógica

Quantificador	Desmembrando	Construindo
$\forall$	Instanciação Universal (**IU**) • Altera a variável ligada para variável livre ou para uma constante.	Generalização Universal (**GU**) • Altera uma variável livre para uma variável ligada (não uma constante). • A variável ligada não deve aparecer antes na prova justificada por IE ou uma PP não descartada.
$\exists$	Instanciação Existencial (**IE**) • Altera uma variável ligada para uma variável livre (não uma constante). • A variável livre não deve aparecer antes na prova.	Generalização Existencial (**GE**) • Altera tanto uma variável livre quanto uma constante para uma variável ligada.

Para Leigos: A série de livros para iniciantes que mais vende no mundo.

Lógica

PARA

LEIGOS®

Lógica para Leigos

por Mark Zegarelli

ALTA BOOKS
EDITORA
Rio de Janeiro, 2013

ISBN: 978-85-5080-414-9

Impresso no Brasil — 1ª Edição, 2013

Produção Editorial Editora Alta Books **Gerência Editorial** Anderson Vieira **Editoria Para Leigos** Daniel Siqueira	**Supervisão Gráfica** Angel Cabeza **Supervisão de Qualidade Editorial** Sergio Luiz de Souza **Supervisão de Texto** Jaciara Lima	**Conselho de Qualidade Editorial** Anderson Vieira Angel Cabeza Jaciara Lima Marco Aurélio Silva Natália Gonçalves Sergio Luiz de Souza	**Design Editorial** Bruna Serrano Iuri Santos	**Marketing e Promoção** Daniel Schilklaper marketing@altabooks.com.br
Equipe Editorial	Brenda Ramalho Claudia Braga Cristiane Santos Danilo Moura	Evellyn Pacheco Juliana de Paulo Juliana Larissa Xavier Licia Oliveira	Livia Brazil Marcelo Vieira Milena Souza Paulo Camerino	Thiê Alves Vanessa Gomes Vinicius Damasceno
Tradução Wendy Campos	**Copidesque** Ronaldo de Lima Netto	**Revisão Técnica** Rogerio M. dos Santos *Doutor em Engenharia (COPPE/UFRJ) e Bacharel em Matemática*	**Revisão Gramatical** Eva Pereira da Rocha Joris Bianca Da Silva	**Diagramação** Diego Oliveira

Dados Internacionais de Catalogação na Publicação (CIP)

Z44l Zegarelli, Mark.
Lógica para leigos / por Mark Zegarelli. ” Rio de Janeiro, RJ : Alta Books, 2013.
384. : il. ; 24 cm. ” (Para leigos)

Inclui índice.
Tradução de: Logic for Dummies.
ISBN 978-85-7608-802-8

1. Lógica. 2. Raciocínio. 3. Lógica matemática. I. Título. II. Série.

CDU 16
CDD 160

Índice para catálogo sistemático:
1. Lógic 16

(Bibliotecária responsável: Sabrina Leal Araujo ” CRB 10/1507)

Rua Viúva Cláudio, 291 – Bairro Industrial do Jacaré
CEP: 20970-031 – Rio de Janeiro – Tels.: (21) 3278-8069/8419 Fax: 21 3277-1253
www.altabooks.com.br – e-mail: altabooks@altabooks.com.br
www.facebook.com/altabooks – www.facebook.com/alt_abooks

Sobre o Autor

Mark Zegarelli é escritor profissional, graduado em Inglês e Matemática pela Rutgers University. Trabalhou por muitos anos com a criação de inúmeros quebra-cabeças de lógica, produziu vasta documentação de softwares e, ocasionalmente, escreveu críticas de livros e filmes. Chegou a trabalhar, durante sua trajetória, como faxineiro, com pintura decorativa e (por dez horas) com vendas para pagar algumas contas. Porém, escrever é o que ele mais gosta.

Mark passa a maior parte do tempo em Long Branch, Nova Jersey, e, esporadicamente, em São Francisco na Califórnia.

Dedicatória

Este livro é dedicado a Mark Dembrowski, com todo meu amor, por seu incansável apoio, encorajamento e sabedoria.

Agradecimentos

Escritores não escrevem, reescrevem — e claro que reescrever é mais fácil quando se pode contar com a ajuda de um time de editores de primeira linha. Meu muito obrigado a Kathy Cox, Mike Baker, Darren Meiss, Elizabeth Rea e Jessica Smith da Wiley Publications pela orientação com olhos de lince. Vocês fizeram com que este livro fosse possível.

Gostaria de agradecer ao Professor Kenneth Wolfe, da St. John's University, Professor Darko Sarenac, da Stanford University, e ao Professor David Nacin, da Paterson University, pelas inestimáveis revisões técnicas, bem como ao Professor Edward Haertel, da Stanford University, pelo encorajamento e assistência. Vocês tornaram este livro melhor.

Agradeço, ainda, ao apoio motivacional de Tami Zegarelli, Michael Konopko, David Feaster, Dra. Barbara Becker Holstein, ao pessoal da Sunset Landing, em Asbury Park, ao Dolores Park Cafe, em San Francisco, e aos QBs. Vocês tornaram este livro prazeroso.

Sumário Resumido

Parte IV: Lógica Quantitativa (LQ)

Parte V: Desenvolvimentos Modernos na Lógica

Parte VI: A Parte dos Dez

Sumário

Capítulo 11: Grandes Suposições com Provas Indiretas e Condicionais

Capítulo 12: Juntando Tudo: Estratégias para Resolver Qualquer Prova

Capítulo 13: Um por Todos e Todos por Um

Capítulo 24: Dez Grandes Nomes na Lógica

Aristóteles (384-322 a.C.)
Gottfried Leibniz (1646-1716)
George Boole (1815-1864)
Lewis Carroll (1832-1898)
Georg Cantor (1845-1918)
Gottlob Frege (1848-1925)
Bertrand Russell (1872-1970)
David Hilbert (1862-1943)
Kurt Gödel (1906-1978)
Alan Turing (1912-1954)

Capítulo 25: Dez Dicas para Ter Sucesso em um Exame de Lógica

Respire
Comece com uma Rápida Olhada em Todo o Exame
Faça um Aquecimento com um Problema Fácil Primeiro
Preencha as Tabelas Verdade Coluna por Coluna
Se Ficar Travado, Anote Tudo o que Conseguir
Se Ficar REALMENTE Travado, Siga Adiante
Se o Tempo For Curto, Termine as Coisas Mais Enfadonhas
Verifique Seu Trabalho
Admita Seus Erros
Fique Até o Doloroso Final

Introdução

Você usa a Lógica diariamente e sequer se dá conta disso. Observe, por exemplo, algumas das ocasiões em que pode utilizá-la:

- ✓ Ao planejar uma noitada com um amigo
- ✓ Ao pedir aumento ou um dia de folga a seu chefe
- ✓ Ao escolher que camisa comprar entre aquelas que gostou
- ✓ Ao explicar a seus filhos por que eles têm que fazer o dever de casa antes de assistir TV

Você usa a Lógica em todos esses momentos para explicar seu raciocínio e fazer com que as outras pessoas vejam as coisas sob a mesma perspectiva que você.

Embora você nem sempre aja com lógica, ela é natural, pelo menos entre os humanos; é uma das grandes justificativas da longa existência dos homens em um planeta repleto de criaturas muito maiores, mais rápidas, mais ferozes e em maior número.

Depois que se conscientizar de que a Lógica já faz parte da sua vida, você poderá vê-la trabalhando (ou *não*) para onde quer que olhe.

Este livro foi escrito para mostrar como a Lógica surge naturalmente em sua vida cotidiana. Ao perceber isso, você poderá aperfeiçoar alguns tipos de raciocínio, aproximando-os de sua essência. A Lógica lhe dá as ferramentas para trabalhar com aquilo que você já sabe (premissas) para chegar a um próximo passo (conclusão). Ela ainda é muito útil para a localização de falhas em argumentos tais como fragilidade, suposições ocultas ou simplesmente um raciocínio confuso.

Sobre Este Livro

A Lógica está entre nós há muito tempo, quase 2.400 anos, e o tempo continua passando! Então, com tantas pessoas (no passado e no presente) pensando e escrevendo sobre Lógica, pode parecer difícil saber por onde começar, mas, não tenha medo, escrevi este livro pensando nisso.

Se você já estiver tendo aulas de introdução à Lógica, poderá usar este livro para complementar seus conhecimentos. Quase tudo daquilo que você está aprendendo nas aulas será explicado aqui de forma simplificada, com muitos exemplos, passo a passo. Se, da mesma forma, estiver apenas interessado em conhecer Lógica, este livro também é um ótimo lugar para começar.

Lógica para Leigos é para qualquer pessoa que queira conhecer Lógica — o que é, de onde veio, por que foi inventada e até mesmo aonde está indo. Se você já está estudando Lógica, encontrará neste livro explicações claras para tudo o que está aprendendo, com muitos exemplos dos tipos de problemas que seu professor irá pedir para resolver. Este livro lhe dará uma visão geral da Lógica em suas muitas formas e uma base sólida de conhecimento em que trabalhar.

A Lógica é uma das poucas áreas de estudo lecionadas em dois departamentos acadêmicos distintos: Matemática e Filosofia. A razão pela qual ela pode se encaixar em duas categorias de estudo aparentemente tão distintas é histórica: a Lógica foi fundada por Aristóteles e desenvolvida por filósofos, durante séculos. Porém, há cerca de 150 anos, os matemáticos descobriram que a Lógica era uma ferramenta indispensável para embasar seus trabalhos à medida que evoluíam para um campo mais e mais abstrato.

Um dos mais importantes resultados dessa sobreposição é a Lógica Formal, que parte das ideias da Lógica Filosófica e as aplica na estrutura da Matemática. A Lógica Formal é geralmente ensinada nos departamentos de Filosofia com um objetivo puramente computacional, isto é, matemático.

Quando escrevi este livro tentei contrabalançar ambos os aspectos da Lógica. Falando de uma forma geral, o livro começa no mesmo lugar que a Lógica — na Filosofia — e termina onde ela foi aplicada: na Matemática.

Convenções Usadas Neste Livro

Para ajudá-lo a navegar por este livro, usarei as seguintes convenções:

- ✓ *Itálico* é usado para dar ênfase e para destacar palavras novas e termos definidos no texto. Ele também é usado para as variáveis nas equações.
- ✓ **Negrito** indica palavras-chave em listas de itens e é usado para identificar verdadeiro "**V**" e falso "**F**" na equações e tabelas. Ele também é utilizado para as 18 regras de inferência na LP e para as 5 regras de inferência na LQ.
- ✓ Box são as caixas acinzentadas que contêm uma informação interessante, mas não crucial para a sua compreensão do capítulo ou do tópico.
- ✓ Texto em negrito e em fonte maior (**V** e **F**) é usado para os exemplos de tabelas verdade e tabelas rápidas para indicar informação que acabou de ser adicionada. É usado em tabelas verdade completas e tabelas rápidas para indicar a veracidade da proposição como um todo.
- ✓ Parênteses são usados nas proposições, ao invés da combinação de parênteses, colchetes e chaves. Veja o exemplo:

  $\sim((P \vee Q) \rightarrow \sim R)$

Só de Passagem

Eu adoraria que você lesse este livro de cabo a rabo, mas vou ser franco: ninguém tem tempo sobrando nos dias de hoje. O quanto você vai ler deste livro vai depender do seu grau de conhecimento e o quanto quer se aprofundar na Lógica.

No entanto, sinta-se à vontade para pular qualquer parte sinalizada com o ícone de Papo de Especialista. Essas informações, embora interessantes, são geralmente muito técnicas e podem ser deixadas de lado. Poderá pular, ainda, todos os *boxes* que encontrar pelo caminho. Esses pequenos apartes geralmente trazem alguma informação histórica ou incomum, mas que não é essencial para a compreensão do Capítulo.

Penso que...

Veja o que penso sobre você:

> ✓ Quer conhecer mais sobre Lógica, seja como estudante ou apenas um curioso
>
> ✓ Sabe distinguir proposições verdadeiras de falsas sobre fatos conhecidos, tais como “Pedro Álvares Cabral descobriu o Brasil” e “O Cristo Redentor está situado no Rio de Janeiro”
>
> ✓ Compreende equações matemáticas simples
>
> ✓ Consegue entender Álgebra Básica e resolve, por exemplo, equações como esta: $7 - x = 5$

Como Este Livro Está Organizado

Este livro está separado em seis partes. Apesar de cada uma das partes ser embasada nas informações contidas nas partes anteriores, o livro está organizado de forma modular. Então, sinta-se à vontade para seguir a ordem que achar melhor. Quando discuto um novo tópico que depende de matéria mais básica, faço referência ao capítulo onde essa matéria foi estudada. Se, por ora, você necessitar apenas de informação sobre determinado tópico, verifique o Índice ou o Sumário — eles lhe indicarão a direção correta.

Veja um breve esquema dos tópicos abrangidos pelo livro:

Parte I: Visão Geral da Lógica

O que é lógica? O que significa pensar logicamente ou ilogicamente e como distingui-los? A Parte I responde a essas perguntas (e muitas mais!). Os capítulos nessa parte discutem a estrutura da argumentação lógica, explicam o que são premissas e conclusões e trilha o desenvolvimento da Lógica em suas muitas formas, desde os gregos até os vulcanos.

Parte II: Lógica Sentencial Formal (LS)

A Parte II é uma introdução à Lógica Formal, também chamada de Simbólica, que usa um conjunto próprio de símbolos para substituir as frases em uma linguagem natural, como o português. A grande vantagem da Lógica Formal é que se trata de uma forma mais fácil e clara de expressar proposições lógicas que poderiam ser longas e complicadas em português (ou em suaíli).

Você descobrirá a Lógica Sentencial (LS) e os cinco operadores lógicos que a compõem. Mostrarei, ainda, como traduzir do português para a Lógica Sentencial e vice-versa. Finalmente, o ajudarei a entender como analisar uma proposição para decidir se é verdadeira ou falsa, usando três ferramentas simples: tabela verdade, tabela rápida e árvores lógicas.

Parte III: Provas, Sintaxe e Semântica na LS

Como qualquer estudioso de Lógica, tenho certeza de que você está morrendo de vontade de saber como escrever provas em LS – isso mesmo, aqueles argumentos formais irritantes que interligam um conjunto de premissas a uma conclusão — usando as regras de inferência. Está com sorte. Nesta parte, descobrirá os detalhes para escrever provas e aprenderá, ainda, as provas condicionais e as indiretas, além de abordá-las da forma mais eficiente possível, usando uma variedade de estratégias.

Passará a encarar a LS de uma perspectiva mais ampla, examinando a Lógica Sentencial nos níveis da sintaxe e da semântica.

Descobrirá como distinguir uma proposição de uma série de símbolos que apenas se parecem com ela. Discutirei como os operadores lógicos na LS permitem a construção das funções sentenciais que tenham um ou mais valores de entrada e um valor de saída. A partir dessa perspectiva, você verá como a LS é versátil para expressar todas as possíveis funções sentenciais com um mínimo de operadores lógicos.

Parte IV: Lógica Quantitativa (LQ)

Se você quer saber tudo sobre a Lógica Quantitativa (ou LQ), achou o lugar certo: essa parte funciona como um pacote introdutório completo. A LQ abrange tudo da LS, mas se estende em inúmeras formas importantes.

Nessa parte, mostrarei como a LQ permite que você entenda algumas dificuldades de uma proposição em português, desmembrando-a em partes menores daquelas que seriam passíveis de desmembramento na LS. Apresentarei os dois operadores quantitativos que possibilitam expressar uma maior variedade de proposições. Finalmente, mostrarei como usar aquilo que você já sabe sobre provas e árvores lógicas na LQ.

Parte V: Desenvolvimentos Modernos na Lógica

O poder e a sutileza da Lógica se tornam evidentes quando você analisa os avanços ocorridos nesse campo no último século. Nessa parte, você verá como a Lógica transformou em realidade o sonho do computador do século XIX. Discuto como as variações da Lógica Pós-clássica, originada em presunções aparentemente ilógicas, podem ser consistentes e úteis para a descrição de eventos do mundo real.

Demonstro, ainda, como os paradoxos desafiaram a essência da Lógica e forçaram os matemáticos a retirarem dela todas as suas ambiguidades, dispondo-a em sistemas de axiomas. Em última análise, os paradoxos inspiraram um matemático a atrelá-los entre si, como uma forma de provar que a Lógica tem suas limitações.

Parte VI: A Parte dos Dez

Todos os livros da série *Para Leigos* contêm a Parte dos Dez. Só por diversão, essa parte inclui algumas listas dos “dez mais” sobre diversos tópicos: ótimas citações, lógicos famosos e dicas para ser aprovado em exames.

ícones Usados Neste Livro

Você encontrará, ao longo deste livro, quatro ícones que indicam diferentes tipos de informação:

Esse ícone é usado para indicar as noções principais que você precisa saber. Certifique-se de ter entendido a informação contida nesses parágrafos antes de continuar.

Esse ícone destaca dicas úteis que lhe mostrarão o caminho mais fácil a seguir. Experimente-as, especialmente se estiver fazendo um curso de Lógica.

Não pule esses ícones! Eles lhe mostrarão os erros mais comuns a serem evitados. Os parágrafos indicados por esse ícone o ajudam a reconhecer as armadilhas antes de você cair nelas.

Esse ícone avisa sobre trivialidades interessantes, mas desnecessárias e, portanto, pode escolher se quer ler ou pular.

De Lá para Cá, Daqui para Lá

Se você já possui algum conhecimento em Lógica e já está familiarizado com os tópicos da Parte I, sinta-se à vontade para pular para a parte onde está a verdadeira ação. Cada uma das partes é um passo à frente em relação à anterior, então, se você conseguir entender a Parte III, sem problemas, provavelmente não terá que se concentrar nas Partes I e II (a menos, é claro, que você queira fazer uma pequena revisão).

Se você já está estudando Lógica, talvez seja importante ler as Partes III e IV com atenção, — e até mesmo tentar reproduzir algumas provas dos capítulos com o livro fechado. É melhor descobrir o que você não sabe enquanto estuda do que enquanto estiver sofrendo para fazer um exame!

Se você não está estudando Lógica — em um curso completo, com um professor, exames e nota —, e só quer aprender suas noções básicas, talvez queira pular ou apenas dar uma lida rápida nos exemplos detalhados de provas das Partes III e IV. Assim, você também terá uma boa noção sobre a Lógica, mas sem muito esforço.

Se você avançar rapidamente para as Partes IV e V, provavelmente está pronto para lidar com algumas ideias mais avançadas. Se está se coçando para atacar a Lógica mais substancial, dê uma olhada no Capítulo 22. Esse Capítulo sobre os paradoxos lógicos tem algumas coisas muito boas para acelerar seu raciocínio a uma velocidade surpreendente. Boa viagem!

Parte I:
Visão Geral da Lógica

A 5ª Onda Por Rich Tennant

"Ria se quiser, mas não me sinto à vontade em dar aula de Lógica à turma sem usar minhas meias da sorte."

Nesta parte...

Deixe-me adivinhar, você acabou de assistir a sua primeira aula de Lógica e está lutando para descobrir seus segredos o mais depressa possível, pois sua primeira prova será daqui a 48 horas. Talvez esteja apenas procurando uma solução mágica para aumentar sua compreensão sobre o assunto. Em ambos os casos, você veio ao lugar certo.

Nesta parte você faz um primeiro contato com a Lógica. O Capítulo 1 lhe traz uma visão geral sobre como usar a Lógica o tempo todo (conscientemente ou não) para transformar tudo aquilo que sabe em uma melhor forma para compreender o mundo. O Capítulo 2 apresenta a história da Lógica, tendo em vista os diversos tipos já inventados ao longo dos séculos. Finalmente, caso esteja ansioso para botar as mãos na massa, dê uma espiada no Capítulo 3 para conhecer a estrutura básica do argumento lógico. Esse capítulo enfoca, ainda, os conceitos-chave, tais como premissas e conclusões, e discute as formas de testar um argumento para verificar sua validade e correção.

Capítulo 1

O que É Essa Coisa Chamada Lógica?

Neste Capítulo

- Vendo o mundo de um ponto de vista lógico
- Usando a Lógica para construir argumentos válidos
- Aplicando as leis do pensamento
- Entendendo a conexão entre Matemática e Lógica

Vivemos em um mundo ilógico. Se duvida disso, dê uma rápida olhada nos jornais ou ouça atentamente as pessoas sentadas ao seu lado no balcão de um bar. Melhor ainda, passe um final de semana com seus sogros.

Com tantas pessoas pensando e agindo de maneira ilógica, por que você seria diferente? Não seria mais sensato ser tão ilógico quanto o resto da humanidade?

Está certo, ser ilógico de propósito provavelmente não é a melhor ideia. Primeiro, como seria possível que o sensato é não ser lógico? Segundo, se você escolheu este livro, certamente não é daquelas pessoas que nasceram para serem ilógicas. Vamos encarar os fatos — algumas pessoas prosperam no caos (ou alegam isso), enquanto outras, não.

Você conhecerá, neste capítulo, os fundamentos da Lógica e como eles se aplicam à sua vida. Apresentarei poucas palavras e ideias que são importantes para a Lógica e mostrarei, rapidamente, as conexões entre Lógica e Matemática.

Assumindo uma Perspectiva Lógica

Quer seja de forma consciente ou não, você já sabe muito sobre Lógica. Na verdade, nasceu com um detector de lógica acoplado. Não acredita? Faça esse pequeno teste para ver se pode ser considerado uma pessoa lógica:

> **P:** Quantas panquecas são necessárias para cobrir uma casinha de cachorro?
>
> **R:** 23, pois bananas não têm ossos.

Se a resposta lhe parece ilógica, é sinal de que está, ao menos, no caminho de ser lógico. Simplesmente porque, se pode identificar o que é ilógico, deve ter percepção suficiente daquilo que, de fato, é lógico.

Neste tópico, começo com o que você *já* entende sobre Lógica (embora nem sempre tenha consciência disso) e prossigo na construção do alicerce que o ajudará a estudá-la.

Atravessando de um ponto a outro

As crianças são, em sua maioria, curiosas por natureza. Estão sempre querendo saber o *porquê* de cada coisa e sempre têm outra *pergunta* para cada *resposta*. Eis algumas delas, típicas das crianças:

> Por que o Sol nasce de manhã?
>
> Por que tenho que ir à escola?
>
> Por que o carro funciona quando você gira a chave?

Por que as pessoas não obedecem a lei mesmo sabendo que podem ir para a cadeia?

Há um grande mistério quando se pensa sobre isso: se o mundo não tem sentido, por que as coisas deveriam tê-lo?

As crianças percebem, desde a mais tenra idade, que mesmo quando não conseguem entender algo, deve haver uma resposta. Pensam: “se estou aqui e a resposta está lá, o que tenho que fazer para chegar até ela?” — a resposta para essa pergunta geralmente é atormentar os pais com mais perguntas.

Passar de um estágio a outro — da ignorância ao conhecimento — é uma das principais razões para o surgimento da Lógica. Ela nasceu da necessidade inata do ser humano de entender o mundo e, tanto quanto possível, controlá-lo.

Entendendo causa e efeito

Uma das formas de entender o mundo é perceber as conexões entre causa e efeito.

À medida que você passa de criança a adulto, começa a compreender que um fato acarreta o outro. Normalmente essas conexões entre causa e efeito podem ser dispostas em *proposições condicionais*. Considere, por exemplo, algumas delas:

> *Se* eu deixar minha bola favorita rolar para debaixo do sofá, *então* eu não conseguirei alcançá-la.
>
> *Se* eu fizer todo o meu dever de casa antes que meu pai chegue do trabalho, *então* ele jogará bola comigo antes do jantar.
>
> *Se* eu treinar sozinho neste verão, *então*, no outono, o treinador me escolherá para o time de futebol.
>
> *Se* eu continuar convidando-a para sair com gentileza e confiança, *então*, um dia, ela aceitará.

Entender como as proposições condicionais funcionam é um importante aspecto da Lógica.

Analisando as proposições condicionais

LEMBRE-SE

Cada proposição condicional é composta de duas proposições menores, chamadas *subproposições*: a *antecedente*, que segue a palavra *se*, e a *consequente*, que segue a palavra *então*. Veja a seguinte proposição condicional:

> *Se* são 17h, *então* é hora de ir para casa

Nessa proposição, a antecedente é a subproposição

> São 17h.

A consequente é a subproposição

> É hora de ir para casa

Observe que essas subproposições continuam sendo proposições completas isoladamente.

Interligando proposições condicionais

Em muitos casos a consequente de uma proposição condicional se torna a antecedente de outra. Quando isso acontece, tem-se uma cadeia de consequências, que os gregos chamavam de *sorites.* Por exemplo:

Se são 17h, então é hora de ir para casa.

↓

Se é hora de ir para casa, então é quase hora do jantar.

↓

Se é quase hora do jantar, então preciso ligar para meu marido para que ele faça as reservas no restaurante.

Nesse caso, essas proposições condicionais podem ser interligadas para formar uma nova proposição condicional:

> Se são 17h, então preciso ligar para meu marido para que ele faça reservas no restaurante.

Engrossando o Caldo

Com mais experiência de vida, pode-se perceber que as conexões entre causa e efeito vão se tornando cada vez mais sofisticadas:

> *Se* eu deixar minha bola favorita rolar para debaixo do sofá, *então* não conseguirei alcançá-la, *a menos que* eu grite bem alto para que minha avó a pegue para mim, *embora se* eu fizer isso mais de uma vez, *então* ela vai se irritar e me colocar de castigo.
>
> *Se* eu treinar por conta própria neste verão, *mas* não muito pesado para que eu não machuque meu joelho, *então*, no outono, o técnico vai me escolher para o time de futebol, *mas somente se* ele tiver uma vaga no time; *no entanto*, *se* eu não treinar, *então* o treinador *não* vai me escolher.

Tudo e mais um pouco

À medida que você começa a entender o mundo, passa a fazer mais proposições gerais sobre ele, por exemplo:

> Todos os cavalos são dóceis.
>
> Todos os garotos são bobos.
>
> Todo professor daquela escola me persegue.
>
> Toda vez que o telefone toca é para minha irmã.

Palavras tais como *todos(as)* e *todo(a)* permitem-nos classificar as coisas em *conjuntos* (grupo de objetos) e *subconjuntos* (grupos dentro de grupos). Por exemplo, quando se diz “Todos os cavalos são dóceis”, significa que o conjunto de todos os cavalos *está contido no* conjunto de todas as coisas dóceis.

A existência por si só

Você também pode descobrir o mundo separando aquilo que *existe* do que *não existe.* Por exemplo:

> Alguns dos meus professores são legais.
>
> Há pelo menos uma garota na escola que gosta de mim.
>
> Ninguém no clube de xadrez consegue me vencer.
>
> Não existem marcianos.

Palavras como *algum(a)(s), há, existe(m)* mostram uma sobreposição de conjuntos chamada de *interseção.* Por exemplo, quando você diz “Alguns dos meus professores são legais”, significa que há uma interseção entre o conjunto dos meus professores e o de coisas legais.

Da mesma forma, palavras como *não, não há, não existe, ninguém ou nenhum(a)* mostram que não há uma interseção entre conjuntos. Por exemplo, quando você diz “Ninguém no clube de xadrez consegue me vencer”, significa que não há interseção entre o conjunto de todos os membros do clube de xadrez e aqueles de todos os jogadores de xadrez que conseguem vencê-lo.

Algumas palavras da Lógica

Como você pode ver, certas palavras dizem muito mais quando se começa a fazer conexões lógicas. Algumas das palavras mais comuns são:

se... então	e	mas	ou
não	a menos que	embora	todo (a)
todos(as)	todo(a)	existe	há
algum(a) não	há/não existe	nenhum(a)	ninguém

Olhar de forma mais atenta para palavras como essas é um importante exercício de Lógica, pois quando se faz isso, pode-se ver como essas palavras permitem dividir o mundo em duas maneiras distintas (e, assim, entendê-lo melhor).

Construindo Argumentos Lógicos

Quando as pessoas dizem “Vamos ser lógicos” em determinada situação ou problema, geralmente querem dizer “Vamos seguir os seguintes passos”:

1. Determinar o que sabemos que é verdade.
2. Passar algum tempo pensando no assunto.
3. Encontrar o melhor plano de ação.

Em termos lógicos, esses três passos envolvem a construção de um *argumento lógico.* Um argumento contém um conjunto de premissas no início e uma conclusão no final. Em muitos casos, as premissas e a conclusão serão interligadas por uma série de passos intermediários. Nos tópicos a seguir, analiso esses passos na ordem em que geralmente são encontrados.

Gerando premissas

Premissas são os fatos que envolvem a questão, as proposições que você sabe (ou acredita firmemente) que sejam verdadeiras. Escrever o conjunto de premissas é, muitas vezes, um bom primeiro passo para a solução do problema.

Por exemplo, suponha que você seja membro do conselho escolar que está deliberando sobre se deve ou não apoiar a construção de uma nova escola para iniciar as aulas em fevereiro. Todos estão muito animados com o projeto, mas você faz alguns contatos telefônicos e junta alguns fatos, ou premissas.

Premissas:

> Os recursos para o projeto só estarão disponíveis em setembro.

> A construtora não iniciará as obras antes de receber o pagamento.
>
> O projeto total demorará pelo menos oito meses para ser concluído.

Até agora você só tem um conjunto de premissas. Porém, quando você as une, está mais próximo do produto final — seu argumento lógico. No próximo tópico, mostro como combinar essas premissas.

Transpondo lacunas com passos intermediários

Um argumento é, algumas vezes, apenas um conjunto de premissas seguido de uma conclusão. Em muitos casos, porém, ele também inclui alguns *passos intermediários* que demonstram como as premissas levam gradativamente à conclusão.

Usando o exemplo da construção da escola referido no tópico anterior, podese organizar os fatos da seguinte forma:

> De acordo com as premissas, não poderemos pagar a construtora até setembro, de modo que ela só concluirá a obra oito meses depois desse mês, ou seja, em maio. Mas a escola deve iniciar as aulas em fevereiro. Portanto...

A palavra *portanto* indica a conclusão e é o início do passo final, que é tratado a seguir.

Chegando a uma conclusão

A *conclusão* é o resultado de um argumento. Se você escreveu os passos intermediários em clara progressão, a conclusão deve ser óbvia. Para o exemplo da construção da escola que estou usando, tem-se:

Conclusão:

> A construção não estará pronta até o início das aulas.

Se a conclusão não for óbvia ou não fizer sentido, algo está errado com seu argumento. Em alguns casos, um argumento pode não ser válido e, em outros, você pode ter esquecido de algumas premissas, que deverão ser acrescentadas.

Decidindo sobre a validade de um argumento

Depois de construir um argumento, você precisa saber se é *válido*, isto é, se ele é bom.

Para testar a validade de um argumento, presuma que todas as premissas sejam verdadeiras e, então, observe se a conclusão decorre automaticamente delas. Na hipótese afirmativa, o argumento é válido; caso contrário, ele é *inválido*.

Entendendo os entimemas

O argumento da construção da escola que foi examinado pode parecer válido, mas também pode gerar algumas dúvidas. Por exemplo, se outra fonte de recursos for disponibilizada, a construtora poderá começar as obras mais cedo e talvez possa concluí-la até fevereiro. Assim, o argumento possui uma premissa oculta chamada de *entimema*, que seria a seguinte:

> Não há outra fonte de recursos para o projeto.

Argumentos lógicos sobre situações reais (em contraposição a argumentos matemáticos ou científicos) quase sempre possuem entimemas. Então, quanto mais entimemas ocultos você puder descobrir em um argumento, melhor será sua chance de determinar de forma segura se ele é válido.

Descobrir premissas ocultas em argumentos do mundo real está mais relacionado à *retórica*, que é o estudo da construção de argumentos cogentes e convincentes. Falarei mais detalhadamente sobre retórica e estrutura dos argumentos lógicos no Capítulo 3.

Simplificando Conclusões Lógicas com as Leis do Pensamento

Como base para entender Lógica, o filósofo Bertrand Russell estabeleceu as três leis do pensamento. Estas são embasadas em ideias datadas da época de Aristóteles, o fundador da Lógica Clássica, há mais de 2.300 anos (veja o Capítulo 2 para saber mais sobre a história da Lógica).

Todas as três leis são muito elementares e de fácil compreensão. Porém, o mais importante a ser assinalado é que elas permitem que cheguemos a conclusões lógicas sobre proposições, mesmo não estando familiarizados com as reais circunstâncias daquilo que está sendo discutido.

A lei da identidade

LEMBRE-SE

A *lei da identidade* estabelece que cada coisa individual é idêntica a si mesma.

Por exemplo:

> Regina Duarte é Regina Duarte.
>
> Meu gato, Téo, é meu gato, Téo.
>
> O Cristo Redentor é o Cristo Redentor.

Sem qualquer outra informação sobre o mundo, você pode saber, com certeza e por pura lógica, que cada uma dessas proposições é verdadeira. A lei da identidade mostra que qualquer proposição no formato “*X* é *X*” tem que ser verdadeira. Em outras palavras, tudo o que existe no universo é igual a si mesmo. No Capítulo 19 você verá como essa lei é explicitamente aplicada na Lógica.

A lei do terceiro excluído

A *lei do terceiro excluído* afirma que toda proposição é verdadeira ou falsa.

Por exemplo, observe estas duas proposições:

> Meu nome é Marcos.
>
> Meu nome é Alberto.

De novo, sem qualquer outra informação sobre o mundo, você pode logicamente afirmar que cada uma dessas proposições é verdadeira ou falsa. Pela lei do terceiro excluído, não há terceira opção possível — em outras palavras, proposições não podem ser parcialmente verdadeiras nem parcialmente falsas. De certa forma, na Lógica, todas as proposições são completamente verdadeiras ou completamente falsas.

Assim, fico satisfeito que a primeira proposição seja verdadeira, e bastante aliviado que a segunda seja falsa.

A lei da não-contradição

A *lei da não-contradição* afirma que quando uma proposição é considerada verdadeira, sua oposta é necessariamente falsa.

Por exemplo:

> Meu nome é Alberto.
>
> Meu nome não é Alberto.

Mesmo que você não saiba meu nome, poderá ter certeza, por pura lógica, que uma dessas proposições é verdadeira e a outra é falsa. Em outras palavras, por causa da lei da contradição, meu nome pode ser como pode não ser Alberto.

Combinando Lógica e Matemática

Ao longo deste livro, diversas vezes demonstrarei meus pontos de vista com exemplos que usam a Matemática. (Não se preocupe — não haverá nada aqui que você não tenha aprendido na quinta série ou antes.) Matemática e Lógica combinam muito bem por duas razões, que analisarei nos tópicos a seguir.

A Matemática é boa para compreender a Lógica

Nas explicações contidas neste livro usarei, algumas vezes, exemplos de coisas obviamente verdadeiras ou falsas para demonstrar meus pontos de vista. Assim, exemplos de Matemática são muito bons nesse papel pois, na Matemática, uma proposição é sempre verdadeira ou falsa, não há meio-termo.

Por outro lado, alguns fatos aleatórios a respeito do mundo podem ser mais subjetivos ou passíveis de controvérsia. Veja, por exemplo, as duas proposições a seguir:

> Lula foi um ótimo presidente.
>
> *As Aventuras de Huckleberry Finn* é um péssimo livro.

A maioria das pessoas provavelmente concordaria que a primeira proposição é verdadeira e que a segunda é falsa mas, nos dois casos, as afirmações podem gerar controvérsias. Observe, agora, as duas proposições a seguir:

> O número 7 é menor do que o número 8.
>
> Cinco é um número par.

Obviamente, não haverá qualquer controvérsia sobre a veracidade da primeira proposição e da falsidade da segunda.

A Lógica é boa para compreender a Matemática

Como já disse anteriormente neste capítulo, as leis do pensamento nas quais a Lógica é baseada, como, por exemplo, a lei do terceiro excluído, dependem de um raciocínio preto e branco, objetivo. Nada é mais preto e branco do que a Matemática. Muito embora você possa achar que áreas como história, literatura, política e artes sejam muito mais divertidas, elas contêm muitos outros tons de cinza, muitas variações.

A Matemática é construída sobre a Lógica, da mesma forma que uma casa é construída sobre suas fundações. Caso esteja interessado em entender melhor essa conexão entre Matemática e Lógica, leia o Capítulo 22, que trata de como a Matemática parte de fatos óbvios, chamados de *axiomas* e, então, usando a Lógica, gera conclusões interessantes e complexas chamadas *teoremas.*

Capítulo 2

Os Desenvolvimentos da Lógica: de Aristóteles ao Computador

Neste Capítulo

- Entendendo as origens da Lógica
- Descobrindo a Lógica Moderna e a Clássica
- Analisando a Lógica do século XX

Quando você pensa sobre como os seres humanos podem ser *ilógicos*, fica surpreso ao descobrir o quanto a Lógica se desenvolveu com o passar do tempo. A seguir, apresento uma lista parcial de algumas das variedades da Lógica existentes por aí, nesse imenso mundo de premissas e conclusões:

Lógica Booleana	Lógica Moderna	Lógica Quantitativa
Lógica Clássica	Lógica Multivalorada	Lógica Quântica
Lógica Formal	Lógica Não Clássica	Lógica Sentencial
Lógica Difusa	Lógica Predicativa	Lógica Silogística
Lógica Informal	Lógica Proposicional	Lógica Simbólica

Ao se deparar com todas essas variações da Lógica, você pode se sentir compelido a abraçar a humanidade por inteiro e deixar a Lógica para os vulcanos. A boa notícia é que, como descobrirá em breve, algumas delas são muito parecidas. Depois que você estiver familiarizado com umas, as demais serão facilmente compreendidas.

Então, de onde surgiram todos esses tipos de Lógica? É uma longa história, na verdade, é uma história que se estende por mais de 2.000 anos. Pode parecer muito tempo para caber em um único capítulo, mas não se preocupe, pois irá conhecer apenas os detalhes mais importantes. Então, prepare-se para nossa breve aula de história.

Lógica Clássica — de Aristóteles ao Iluminismo

Os gregos antigos tiveram alguma participação no descobrimento de quase tudo, e a Lógica não é uma exceção. Por exemplo, Tales e Pitágoras aplicaram argumentos lógicos na matemática. Sócrates e Platão aplicaram formas semelhantes de raciocínio às questões filosóficas. Porém, o verdadeiro fundador da Lógica Clássica foi Aristóteles.

Quando falo de *Lógica Clássica* neste tópico, estou me referindo ao período histórico em que a Lógica se desenvolveu em contraposição à *Lógica Moderna*, que será examinada mais adiante, ainda neste capítulo. Porém, Lógica Clássica também pode significar o tipo de Lógica mais comum (que é a tratada na maior parte deste livro), em contraposição à *Lógica Não Clássica* (examinada no Capítulo 21). Tentarei ser bem claro ao tratar dessa diferença.

Aristóteles inventa a Lógica Silogística

Antes de Aristóteles (384-322 a.C.), o argumento lógico era aplicado intuitivamente e quando conveniente, na Matemática, nas Ciências e na Filosofia. Por exemplo, considerando que todos os números são pares ou ímpares, se você for capaz de demonstrar que um deles não é par, saberá que ele só pode ser ímpar. Os gregos se sobressaíram usando essa abordagem de dividir e conquistar. Eles usavam a lógica regularmente, como uma ferramenta para analisar o mundo.

Aristóteles, porém, foi o primeiro a reconhecer que a ferramenta em si poderia ser examinada e desenvolvida. Em seis trabalhos sobre lógica — mais tarde compilados em um único trabalho, chamado *Órganon*, que significa *instrumento* —, ele estudou como um argumento lógico funciona. Aristóteles esperava que a Lógica, sob sua nova formulação, servisse como uma ferramenta do pensamento, que ajudaria os filósofos a entenderem melhor o mundo.

Aristóteles considerava que o objetivo da Filosofia era ser um conhecimento científico, e achava que a estrutura desse conhecimento estava na Lógica. Usando a Geometria como modelo, viu que a ciência consistia de provas, provas de silogismo, silogismos de proposições e proposições de termos. Então, no *Órganon*, Aristóteles trabalhou de baixo para cima: o primeiro livro, *Categorias*, trata dos termos; o segundo, *Da Interpretação*, das proposições; o terceiro, *Primeiros Analíticos*, dos silogismos; e o quarto, *Segundos Analíticos*, das provas.

Primeiros Analíticos, o terceiro livro da série *Órganon*, analisa diretamente o que Aristóteles chamava de *silogismos*, que são estruturas de argumentos que, por suas próprias formas, parecem ser incontestavelmente válidos.

A ideia por trás do silogismo era simples, tão simples na verdade, que ela fora subestimada pelos filósofos e matemáticos, até que Aristóteles a notou.

Em um silogismo, as premissas e conclusões se encaixam de tal forma que, uma vez que você aceita as premissas como verdadeiras, fica obrigado a aceitar que a conclusão também o é, independentemente do teor do real argumento que está sendo construído.

Como exemplo, observe o mais famoso silogismo de Aristóteles:

Premissas:

Todos os homens são mortais.

Sócrates é homem.

Conclusão

Sócrates é mortal.

O argumento a seguir é muito semelhante ao primeiro, quanto à forma; e é a forma do argumento, não seu conteúdo, que o torna incontestável. Uma vez que você aceita as premissas como verdadeiras, a conclusão que se segue é igualmente verdadeira.

Premissas:

Todos os palhaços são assustadores.

Bobo é um palhaço.

Conclusão:

Bobo é assustador.

Classificando as proposições categóricas

Grande parte da atenção de Aristóteles se concentrou no entendimento daquilo que ele chamava de *proposições categóricas*, que eram simples proposições que falavam sobre categorias inteiras de objetos ou pessoas. Móveis, cadeiras, pássaros, árvores, coisas vermelhas, filmes de Meg Ryan e cidades cujos nomes começam com a letra *T* são exemplos de categorias.

De acordo com a lei do terceiro excluído (discutida no Capítulo 1), tudo pode ser incluído dentro ou fora de uma determinada categoria. Por exemplo, uma cadeira vermelha está inserida na categoria dos móveis, das cadeiras e das coisas vermelhas, mas não na categoria dos pássaros, das árvores, dos filmes da Meg Ryan ou das cidades cujos nomes começam com a letra *T.*

Aristóteles dividiu as proposições categóricas em dois tipos:

- **Proposições universais:** São aquelas que afirmam algo sobre toda a categoria. Veja um exemplo de proposição universal:

  Todos os cachorros são leais.

  Essa proposição relaciona duas categorias e afirma que tudo o que está contido na categoria de cachorros também está contido na categoria de coisas leais. Você pode considerar que essa é uma proposição universal, pois ela afirma que lealdade é uma qualidade universal dos cachorros.

- **Proposições particulares:** São as que afirmam a existência de pelo menos um exemplo dentro de uma categoria. Observe a proposição particular a seguir:

  Alguns ursos são perigosos.

  Essa proposição afirma que ao menos um item da categoria dos ursos pertence também à categoria das coisas perigosas. Ela é considerada uma proposição particular, pois afirma que pelo menos um urso em particular é perigoso.

Entendendo o quadrado das oposições

O *quadrado das oposições* — instrumento desenvolvido por Aristóteles para estudar as proposições categóricas — organiza as quatro formas básicas de proposições categóricas que frequentemente aparecem nos silogismos. Essas quatro formas têm como base as versões afirmativas e negativas das proposições universais e particulares.

Aristóteles organizou esses quatro tipos de proposições em um quadro semelhante ao apresentado na Tabela 2-1. O exemplo mais famoso de Aristóteles foi baseado na proposição “todos os humanos são mortais”. Porém, o exemplo apresentado na tabela a seguir foi inspirado em um gato dormindo.

Tabela 2-1: O Quadrado das Oposições

	Formas Afirmativas	Formas Negativas
Formas Universais	**A:** Todos os gatos estão dormindo.	**E:** Nenhum gato está dormindo
	Não existe um gato que não esteja dormindo	Todos os gatos não estão dormindo
	Nenhum gato não está dormindo.	Não existe um gato que esteja dormindo.
	Todo gato está dormindo.	Não existe um gato dormindo.
Formas Particulares	**I:** Alguns gatos estão dormindo.	**O:** Nem todos os gatos estão dormindo.
	Nem todos os gatos não estão dormindo.	Alguns gatos não estão dormindo.
	Ao menos um gato está dormindo.	Há ao menos um gato que não está dormindo.
	Há um gato dormindo.	Nem todo gato está dormindo.

Como você pode ver na tabela, cada tipo de proposição expressa uma relação diferente entre a categoria dos gatos e a das coisas que estão dormindo. Em português, você pode expressar cada tipo de proposição de inúmeras maneiras. Apenas algumas dessas maneiras foram enumeradas na tabela, mas muitas mais são possíveis em cada um dos casos.

Aristóteles percebeu a relação entre todos esses tipos de proposições. A mais importante delas é a de *contradição* entre as proposições que estão na diagonal umas das outras. Nos pares contraditórios, uma proposição é verdadeira e a outra é falsa.

Por exemplo, veja as proposições **A** e **O** na Tabela 2-1. Obviamente, se todo gato do mundo está dormindo naquele momento, então **A** é verdadeira e **O** é falsa, caso contrário, a situação se inverteria. Da mesma forma, veja as proposições **E** e **I**, se todo gato no mundo está acordado, então **E** é verdadeira e **I** é falsa, caso contrário, a situação se inverteria.

Caso esteja curioso, as letras **A** e **I** usadas para designar as formas positivas teriam surgido da palavra em latim **A***ff***I***rmo*, que significa “eu afirmo”. Do mesmo modo, as letras **E** e **O** teriam se originado da palavra em latim *n**E**g**O***, que significa “eu nego”. A origem dessas designações não é certa, mas podese descartar Aristóteles, que falava grego e não latim.

Os axiomas e teoremas de Euclides

Apesar de Euclides (*ca.* 325-265 a.C.) não ter sido exatamente um lógico, suas contribuições para a Lógica foram inegáveis.

Euclides é mais conhecido por seu trabalho no campo da Geometria, que ainda é chamada *Geometria Euclidiana* em sua homenagem. Sua maior realização foi a organização lógica dos princípios da Geometria em *axiomas* e *teoremas.*

Euclides começou com cinco axiomas (também chamados de *postulados*), proposições verdadeiras que ele acreditava serem simples e óbvias. A partir desses axiomas, ele utilizou a Lógica para demonstrar seus teoremas, proposições verdadeiras que eram mais complexas e não tão óbvias. Por esse caminho, conseguiu demonstrar grande parte da Geometria, logicamente partindo apenas dos cinco axiomas. Matemáticos ainda utilizam essa organização lógica de proposições em axiomas e teoremas. Para saber mais sobre esse tópico, veja o Capítulo 22.

Ele ainda usou uma metodologia lógica, chamada de *prova indireta*. Por esse método, você parte do oposto daquilo que quer provar e, então, demonstra que aquela presunção o leva a uma conclusão obviamente incorreta.

Por exemplo, um detetive investigando um misterioso homicídio pode ponderar: “Se o mordomo cometeu o assassinato, então, ele tem que ter estado na casa entre 7h e 8h da noite. Mas, testemunhas o viram nesse horário em uma cidade a trinta e dois quilômetros da casa, de forma que ele não poderia estar também na casa. Portanto, o mordomo não cometeu o crime”.

A prova indireta é também chamada de *prova por contradição* ou *reductio ad absurdum*, que significa *reduzido ao absurdo.* Veja o Capítulo 11 para saber mais sobre como utilizar a prova indireta.

Crisipo e os estoicos

Enquanto os sucessores de Aristóteles desenvolviam seu trabalho sobre o silogismo lógico de proposições categóricas, os estoicos, que formavam outra escola grega de filosofia, abordavam uma perspectiva diferente. Eles se concentraram no estudo das *proposições condicionais*, que são aquelas que contêm a fórmula *se... então...*. Por exemplo:

> *Se* nuvens estão se formando no oeste, *então* vai chover.

O mais notável dentre esses lógicos foi Crisipo (279-206 a.C.). Ele analisou argumentos usando proposições na forma de se... então....

Por exemplo:

Premissas:

Se nuvens estão se formando no oeste, então vai chover.

Nuvens estão se formando no oeste.

Conclusão:

Vai chover.

Há, certamente, muitas conexões entre a abordagem aristotélica e a estoica. Ambas enfocavam conjuntos de premissas contendo proposições que, quando verdadeiras, tendiam a se encaixar de tal maneira que, obrigatoriamente, levavam a uma conclusão também verdadeira. Mas o atrito entre as duas escolas de pensamento fez com que a Lógica se desenvolvesse, por mais de um século, em duas correntes separadas que, algum tempo depois, todavia, acabaram se fundindo em uma única disciplina.

A Lógica tira férias

Depois dos gregos, a Lógica saiu de férias por um longo período, com alguns renascimentos esporádicos.

Durante o Império Romano e a Europa Medieval, a Lógica foi reiteradamente negligenciada por mais de mil anos. Os trabalhos de Aristóteles sobre lógica eram ocasionalmente traduzidos, com algumas observações do tradutor. No entanto, muito poucas pessoas escreveram tratados sobre lógica.

No primeiro milênio d.C., mais trabalhos sobre lógica foram escritos no mundo árabe. Tanto os filósofos cristãos quanto os muçulmanos, em Bagdá, continuaram a traduzir e comentar os trabalhos de Aristóteles. Avicena (980-1037), contrariou essa prática estudando os conceitos lógicos envolvendo tempo, tais como *sempre, algumas vezes* e *nunca.*

O século XII viveu um renascimento do interesse pela Lógica, especialmente pelas *falácias lógicas*, que são falhas nos argumentos. Alguns desses trabalhos, que foram iniciados por Aristóteles em sua obra *Refutações Sofísticas*, foram utilizados pelos teólogos durante esse período de crescente influência católica na Europa. Durante os séculos seguintes, os filósofos continuaram a estudar as questões sobre a linguagem e os argumentos, quando relacionados à Lógica.

Além disso, como uma das sete artes liberais, a Lógica também era o centro do currículo das universidades que surgiram nessa época. (tenho certeza de que você está morrendo de vontade de saber quais as outras seis eram: Gramática, Retórica, Aritmética, Geometria, Astronomia e Música).

Lógica Moderna — os Séculos XVII, XVIII e XIX

Na Europa, enquanto a Era da Fé gradualmente dava lugar à Era da Razão, nos séculos XVI e XVII, os pensadores ficavam otimistas sobre as respostas às questões envolvendo a natureza e o universo.

Muito embora cientistas (como Isaac Newton) e filósofos (como René Descartes) continuassem a crer em Deus, passaram a procurar a resposta sobre o funcionamento do universo divino além dos ensinamentos da igreja. Ao invés disso, descobriram que muitos dos mistérios do mundo — tais como a queda de uma maçã ou o movimento da lua no espaço — poderiam ser explicados segundo as leis da mecânica e previstos mediante o uso da matemática. Com isso, a flor da lógica surgiu no pensamento científico para se estabelecer como uma ferramenta essencial da razão.

Leibniz e a Renascença

Gottfried Leibniz (1646-1716) foi o maior lógico da Renascença na Europa. Como Aristóteles, Leibniz vislumbrou o potencial da Lógica para se tornar um instrumento indispensável para a compreensão do mundo. Ele foi o primeiro lógico a levar o trabalho de Aristóteles um passo adiante, ao transformar as proposições lógicas em símbolos que poderiam, então, ser manipulados como números e equações. O resultado desse trabalho foi a primeira experiência rudimentar de *Lógica Simbólica.*

Dessa forma, Leibniz esperava que a Lógica pudesse transformar a filosofia, a política e até mesmo a religião em puro cálculo, fornecendo um método confiável para responder a todos os mistérios da vida com objetividade. Em uma famosa citação de *The Art of Discovery* (1685), ele diz:

> A única forma de retificarmos nossos argumentos é torná-los tão tangíveis quanto aqueles dos matemáticos, para que possamos encontrar nossos erros com uma rápida olhada e,

> quando houver disputas entre pessoas, possamos simplesmente dizer: “Vamos calcular, sem mais confusão, para ver quem tem razão.”

Infelizmente, o sonho de Leibniz de transformar todas as áreas da vida em mero cálculo não foi alcançado pela geração que o sucedeu. Suas ideias estavam tão à frente de seu tempo, que seus sucessores nem mesmo compreenderam sua importância. Depois de sua morte, seus trabalhos sobre lógica como um cálculo simbólico acumulavam poeira por quase 200 anos. Quando esses trabalhos foram redescobertos, os lógicos já haviam alcançado e suplantado suas ideias. Como resultado, Leibniz não foi considerado tão influente como deveria ter sido durante essa fase crítica do desenvolvimento da Lógica.

Estimulando a Lógica Formal

A Lógica foi, em sua maior parte, estudada *informalmente* — isto é, sem o uso de símbolos no lugar de palavras — até o início do século XIX. Começando com Leibniz, matemáticos e filósofos dessa época construíram uma ampla variedade de anotações para conceitos lógicos comuns. Esses sistemas, porém, careciam, em geral, de qualquer método para computação de escala completa e cálculo.

Entretanto, no final do século XIX, matemáticos desenvolveram a *Lógica Formal* — também chamada de *Lógica Simbólica* —, na qual símbolos computáveis substituem palavras e proposições. Os três principais contribuidores da Lógica Formal foram George Boole, Georg Cantor e Gottlob Frege.

Álgebra Booleana

Batizada por seu inventor, George Boole (1815-1864), a Álgebra Booleana é o primeiro sistema totalmente detalhado que lida com a

Lógica como cálculo. Por essa razão, tal sistema é considerado o precursor da Lógica Formal.

LEMBRE-SE

A Álgebra Booleana é semelhante à Aritmética padrão no uso dos valores numéricos e de operações familiares tais como adição e multiplicação. Ao contrário da Aritmética, porém, somente dois números são usados: 0 e 1, que significam *falso* e *verdadeiro*, respectivamente.

Por exemplo:

> Considere A = Dom Pedro I declarou a Independência.
>
> Considere B = Paris Hilton escreveu a Constituição Brasileira.

Tendo em vista que a primeira proposição é verdadeira e a segunda é falsa (ainda bem!), você pode dizer que:

> $A = 1$ e $B = 0$

Na Álgebra Booleana, a adição é interpretada como *ou*, então a proposição

> Dom Pedro I declarou a Independência *ou* Paris Hilton escreveu a Constituição Brasileira

é traduzida como

$$A + B = 1 + 0 = 1$$

Tendo em vista que a equação Booleana resulta em 1, a proposição é verdadeira. De modo semelhante, a multiplicação é interpretada como *e*, então a proposição

> Dom Pedro I declarou a Independência *e* Paris Hilton escreveu a Constituição Brasileira

é traduzida como

$$A \times B = 1 \times 0 = 0$$

Nesse caso, a equação Booleana resulta em 0, então a proposição é falsa.

Como podemos ver, o cálculo dos valores é bastante parecido à aritmética. Mas, o significado por trás dos números é pura lógica.

Veja o Capítulo 14 para saber mais sobre a Álgebra Booleana.

Teoria dos conjuntos de Cantor

A *teoria dos conjuntos*, originalmente estudada por Georg Cantor, nos anos 1870, foi outro prenúncio da Lógica Formal, mas com uma influência muito mais ampla e útil do que a Álgebra Booleana.

Em uma definição vaga, um *conjunto* é apenas uma coleção de coisas que podem ou não ter algo em comum. Veja alguns exemplos:

> {1, 2, 3, 4, 5, 6}
>
> {Batman, Mulher-Maravilha, Homem-Aranha}
>
> {África, Kelly Clarkson, Novembro, Snoopy}

Essa simples construção é tremendamente eficiente para caracterizar importantes ideias centrais da Lógica. Por exemplo, veja esta proposição:

> Os nomes de todos os estados que compõem o Brasil que contêm a letra *z* começam com a letra *A.*

Essa proposição pode ser comprovada através da identificação dos conjuntos compostos pelos nomes de todos os estados que contêm a letra *z* e os que começam com a letra *A*. Estes são, respectivamente, os conjuntos de 1 e 2 dados abaixo:

> **Conjunto 1:** {Amazonas} **Conjunto 2:** {Alagoas, Amapá, Amazonas, Acre}

Como você pode ver, todo membro do primeiro conjunto também faz parte do segundo. Assim, o primeiro é um *subconjunto* do segundo, então, a proposição original é verdadeira.

Apesar de sua aparente simplicidade — e, até mesmo, por causa dela — a Teoria dos Conjuntos se tornou o alicerce da Lógica e, posteriormente, da própria matemática formal.

A Lógica Formal de Frege

Gottlob Frege (1848-1925) inventou o primeiro sistema real de Lógica Formal, que consistia em um sistema lógico dentro de outro. O sistema menor, a *Lógica Sentencial* — também conhecida como *Proposicional* —, usa letras para substituir proposições simples, que são interligadas usando os símbolos para os cinco conceitos chaves: *não, e, ou, se*, e *se e somente se.* Por exemplo:

Considere E = Evelyn está no cinema.

Considere P = Pedro está em casa.

Essas definições permitem que você formule tais proposições:

Evelyn está no cinema *e* Pedro está em casa.

Se Evelyn está no cinema, *então* Pedro *não* está em casa.

e transforme-as em símbolos, da seguinte forma:

$E \wedge P$

$E \rightarrow \sim P$

Na primeira proposição, o símbolo ^ significa *e.* Na segunda, o símbolo → significa *se... então* e o símbolo ~ significa *não.*

A Lógica Sentencial será mais detalhadamente estudada nas Partes II e III.

O sistema maior, *Lógica Quantitativa* — também chamada de *Lógica Predicativa* —, inclui todas as regras da Lógica Sentencial, mas vai além.

A Quantitativa usa diferentes letras para substituir o sujeito e o predicado de uma proposição simples. Por exemplo:

Considere e = Evelyn

Considere p = Pedro

Considere Mx = x está no cinema

Considere Hx = x está em casa

Essas definições permitem que você represente tais proposições:

Evelyn está no cinema *e* Pedro está em casa

Se Evelyn está no cinema, *então* Pedro *não* está em casa

Como

$Me \wedge Hp$

$Me \rightarrow \sim Hp$

A Lógica Quantitativa também inclui mais dois símbolos para *todos* e *alguns*, que lhe permitem representar as proposições

Todos estão no cinema

Alguns estão no cinema

Como

$\forall x[Mx]$

$\exists x[Hx]$

A Lógica Quantitativa tem o poder de representar os quatro tipos de proposições categóricas básicas do quadro das oposições de Aristóteles (veja o tópico "Classificando as proposições categóricas" no início deste capítulo). Na verdade, em suas formas mais complexas, a Lógica Quantitativa é tão poderosa quanto todas as formulações de Lógica anteriores.

Veja a Parte IV para saber mais sobre Lógica Quantitativa.

A Lógica a Partir do Século XX

No final do século XIX, seguindo o exemplo de Euclides (veja o tópico “Os axiomas e teoremas de Euclides”, no início deste capítulo), estudiosos buscaram reduzir tudo o que havia na matemática a um conjunto de teoremas logicamente dependentes de apenas alguns axiomas.

Frege, o inventor do primeiro sistema real de Lógica Formal, vislumbrou a possibilidade de a própria matemática ter surgido da Lógica e da Teoria dos Conjuntos. Começando com apenas alguns axiomas sobre conjuntos, ele demonstrou que números e, posteriormente, toda a matemática resultavam da forma lógica desses axiomas.

A teoria de Frege parecia funcionar bem até que Bertrand Russell (1872-1970) descobriu o *paradoxo*, uma inconsistência que ocorria quando um conjunto poderia conter a si próprio como membro. Dentro do sistema de Frege, Russell observou que era possível criar um conjunto que contivesse todos os conjuntos que *não* contivessem a si mesmos como membros. O problema aqui é que se esse conjunto contém a si mesmo, então ele não contém a si mesmo, e vice-versa. Essa inconsistência tornou-se conhecida como o *Paradoxo de Russell* (o Paradoxo de Russel será estudado no Capítulo 22).

Frege ficou arrasado com seu erro, mas Russell conseguiu visualizar o mérito do seu trabalho. De 1910 a 1913, Bertrand Russell e Alfred North Whitehead produziram a obra de três volumes denominada *Principia Mathematica*, um reestudo das ideias de Frege fundamentando a Matemática em axiomas da Teoria dos Conjuntos e da Lógica.

Lógica Não Clássica

O projeto para reduzir a Matemática e a Lógica a uma pequena lista de axiomas revela uma questão interessante: o que acontece se você começar por um diferente conjunto de axiomas?

Uma possibilidade, por exemplo, é permitir que uma proposição seja outra coisa diferente de verdadeira ou falsa. Em outras palavras, uma proposição poderia violar a lei do terceiro excluído (veja o Capítulo 1). A violação ostensiva dessa lei seria impensável para os gregos, mas, com a Lógica formulada simplesmente como um conjunto de axiomas, a possibilidade se torna viável.

Em 1917, Jan Lukasiewicz desbravou a primeira *Lógica Multivalorada*, em que uma proposição pode ser não apenas *verdadeira* ou *falsa*, mas também *possível*. Esse sistema seria útil para a classificação de uma proposição como esta:

> No ano de 2162, o Brasil vencerá a Copa do Mundo.

A introdução da categoria *possível* ao panteão do *verdadeiro* ou *falso* foi o primeiro grande distanciamento da *Lógica Clássica* — todo estudo nessa área existente até então — na direção de uma nova área denominada *Lógica Não Clássica* (você saberá mais sobre ela, incluindo a Lógica Difusa e a Lógica Quântica, no Capítulo 21).

A prova de Gödel

A obra *Principia Mathematica*, série de três volumes escrita por Bertrand Russell e Alfred North Whitehead, estabeleceu firmemente a Lógica como um alicerce essencial da Matemática. No entanto, mais surpresas estavam por vir.

Com a Matemática definida em termos de conjuntos de axiomas surgiu a questão sobre a *consistência* e a *completude* desse novo sistema. Isto é, seria possível utilizar a Lógica para fazer com que toda a proposição verdadeira sobre matemática se originasse desses axiomas, sem nenhuma proposição falsa?

Em 1931, Kurt Gödel demonstrou que um número infinito de proposições matemáticas são verdadeiras, mas não podem ser demonstradas por causa dos axiomas contidos na *Principia.* Ele também demonstrou que qualquer tentativa de reduzir a matemática a um sistema consistente de axiomas produz o mesmo resultado: um número infinito de verdades matemáticas, chamadas de *proposições indecidíveis*, que não podem ser comprovadas pelo sistema.

Esse resultado, chamado de *Teorema da Incompletude*, firmou Gödel como um dos maiores matemáticos do século XX.

LEMBRE-SE

De certo modo, o Teorema da Incompletude de Gödel respondeu aos anseios de Leibniz de que a Lógica pudesse algum dia fornecer um método para calcular as respostas para todos os mistérios da vida. A resposta, infelizmente, foi um "não" definitivo. Se a Lógica, ao menos na formulação da época, era insuficiente para provar todas as verdades matemáticas, o que dizer das verdades do nosso mundo tão complexo?

A era dos computadores

Ao invés de enfocar aquilo que a Lógica não pode fazer, matemáticos e cientistas descobriram infinitas maneiras de usá-la. O mais notável dentre esses usos é o computador, que alguns especialistas (principalmente cientistas da computação) consideram uma das maiores invenções do século XX.

O hardware — projeto físico do sistema de circuitos do computador — usa as *portas lógicas*, que mimetizam as funções básicas da Lógica Sentencial, recebendo dados na forma de corrente elétrica de uma ou duas fontes e fornecendo dados somente sob determinadas condições.

Por exemplo, uma porta *NOT* (NÃO) fornece dados somente quando não há corrente fluindo de uma fonte específica. Uma porta *AND* (E) fornece dados somente quando a corrente está fluindo de ambas as fontes. E, finalmente, uma porta *OR* (OU) fornece dados somente quando a corrente está fluindo de pelo menos uma das duas fontes.

O *software* — programas que direcionam as ações do *hardware* — é todo escrito em *linguagem de computação*, tal como Java, C++, Visual Basic, Ruby ou Python. Embora todas as linguagens de computação tenham suas diferenças, cada uma contém um núcleo de similaridades, incluindo um conjunto de palavras-chave oriundas da Lógica Sentencial, como *e, ou, se... então* entre outros.

Veja o Capítulo 20 para saber mais sobre como a Lógica é usada no *hardware* e no *software* do computador.

Em busca da última fronteira

Será possível que a Lógica algum dia seja suficiente para descrever todas as sutilezas da mente humana e as complexidades do universo? Meu palpite é que provavelmente isso não ocorrerá, especialmente se você considerar sua atual configuração.

No entanto, a Lógica é um instrumento absurdamente poderoso e seus usos apenas começaram a ser aproveitados. E quem sabe se enquanto escrevo este livro os estudiosos continuam trabalhando para desenvolver mais sistemas lógicos importantes para expandir as capacidades da matemática e das ciências? Seus esforços podem ainda produzir invenções que ultrapassem todas as atuais expectativas e sonhos.

Capítulo 3

Apenas para Fins de Argumentação

Neste Capítulo

- Desconstruindo as partes de um argumento lógico
- Separando a Lógica de seus impostores
- Examinando o amplo espectro de aplicações da Lógica

De forma simplificada, Lógica é o estudo de como se distinguir entre um bom e um mau argumento.

Na vida cotidiana, a maioria das pessoas usa a palavra *argumento* para expressar qualquer coisa, desde uma discussão até os fundamentos que embasam uma opinião. Um argumento, porém, não precisa ser sempre uma discussão acalorada ou cheia de raiva. A ideia por trás do *argumento lógico* é simples: quero convencê-lo de algo e, para isso, exponho alguns fatos com os quais você já concordou. A partir daí passo a demonstrar-lhe como aquilo que estou tentando provar decorre naturalmente daqueles fatos.

Neste capítulo explico o que é Lógica e exponho os elementos de um argumento lógico. Também forneço vários exemplos, mostro-lhe o que *não é* Lógica e dou uma rápida olhada nos vários campos que a utilizam. Quando terminar este capítulo, pode ser que você ainda se veja do lado errado de um argumento, mas ao menos não poderá ser acusado de estar sendo ilógico.

Definindo a Lógica

Aqui está o que você precisa saber sobre Lógica:

- ✓ Lógica é o estudo da *validade do argumento*, que significa avaliá-lo, classificando-o como *válido* (bom) ou *inválido* (mau).
- ✓ Um *argumento* na Lógica é um conjunto de uma ou mais *premissas* seguidas de uma *conclusão*, em geral conectada a uma ou mais *proposições intermediárias.*
- ✓ As premissas e a conclusão são sempre *proposições* — frases que trazem uma informação e que podem ser classificadas como verdadeiras ou falsas.
- ✓ Em um *argumento válido*, se todas as premissas são verdadeiras, a conclusão também tem que ser verdadeira.

LEMBRE-SE

Quando junta todas essas informações você tem o seguinte:

> Lógica é o estudo de como decidir em que circunstâncias um conjunto de premissas verdadeiras leva a uma conclusão também verdadeira.

É isso! Mantenha essa definição em mente enquanto estiver lendo este livro. Escreva-a em um papel e cole no espelho do banheiro. Todos os tópicos da Lógica, de uma maneira ou de outra, têm relação direta com essa ideia central.

Examinando a estrutura do argumento

Um *argumento lógico* deve ter uma ou mais premissas seguidas por uma conclusão. Aqui está um exemplo de argumento lógico:

> *Danilo:* Eu te amo.

Mariana: Sim, eu sei.

Danilo: E você me ama.

Mariana: Verdade.

Danilo: E as pessoas que se amam devem se casar.

Mariana: Certo.

Danilo: Então devemos nos casar.

Esse pode não ser o pedido de casamento mais romântico que você já viu, mas dá para entender a ideia. Se Mariana realmente concorda com todas as três proposições de Danilo, seu argumento deve convencê-la a se casar com ele.

Analisando a estrutura do argumento de Danilo mais de perto, você pode ver que ele contém três premissas e uma conclusão. Resumindo o argumento à sua forma mais básica, Danilo está dizendo que:

Premissas:

Eu te amo.

Você me ama.

Pessoas que se amam devem se casar.

Conclusão:

Nós devemos nos casar.

LEMBRE-SE

As premissas e a conclusão de um argumento têm sempre uma coisa em comum: elas são proposições. Uma *proposição* é simplesmente uma frase que traz alguma informação, um pensamento completo.

Por exemplo, as frases a seguir são todas proposições, embora nenhuma delas possa ser enquadrada na categoria de premissa ou conclusão:

- A capital de Minas Gerais é Belo Horizonte.
- Dois mais dois é igual a cinco.
- Seu vestido vermelho é mais bonito do que seu vestido azul.
- Homens são como cachorros.

Em contrapartida as frases a seguir *não* são proposições:

- Um grande Cadilac azul. (Não é uma frase completa)
- Você vem sempre aqui? (Uma pergunta)
- Limpe seu quarto agora mesmo. (Uma ordem)
- Deus do céu! (Uma exclamação)

LEMBRE-SE

Na Lógica, a informação fornecida por uma proposição, e portanto, a própria proposição, pode ser verdadeira ou falsa (essa regra se aplica quer ela esteja sendo usada como uma das premissas ou como a conclusão de um argumento). Isso é chamado de *valores lógicos* da proposição.

Os estudiosos lidam com valores lógicos o tempo todo e tendem a ser um pouco econômicos na hora de se referirem a ele, chamando-o apenas de *valor* da proposição. Uso os dois termos neste livro, mas sempre com o mesmo significado.

Algumas vezes você pode facilmente identificar qual é o valor lógico de uma proposição. O valor de uma proposição que declara “A capital de Minas Gerais é Belo Horizonte” é *verdadeiro*, pois a capital de Minas Gerais é de fato Belo Horizonte. Já dois mais dois é igual a quatro e não cinco, então o valor da proposição “Dois mais dois é igual a cinco” é *falso*.

Em outros casos o valor lógico das proposições é mais difícil de ser identificado. Por exemplo, como decidir se um vestido é realmente mais bonito do que o outro, ou se os homens são realmente iguais a cachorros?

Não se preocupe por ora em *como* descobrir se uma proposição é verdadeira ou falsa, ou mesmo *se* isso pode ser descoberto. Tratarei dessa questão no tópico “A solidez da coerência”, mais adiante neste capítulo.

Em busca de validação

Em um bom argumento — ou *argumento válido* como dizem os lógicos —, quando todas as premissas são verdadeiras, a conclusão também tem que ser verdadeira.

Argumentos válidos são a essência da Lógica. Lembre-se de que em um argumento lógico quero convencê-lo de algo e para isso apresento fatos com os quais você já concordou (*premissas*), e então mostro o que estou tentando provar (a *conclusão*), que decorre naturalmente desses fatos. Se o argumento é válido e incontestável a conclusão, inevitavelmente decorre das premissas.

Por exemplo, suponha que seu professor lhe diga: “Todos os que estudaram foram bem no exame. Você estudou então você foi bem.” Desmembre essa proposição em premissas e conclusão para ver aonde chega:

Premissas:

Se um aluno estudou, então ele foi bem no exame.

Você estudou.

Conclusão:

Você foi bem no exame.

O argumento anterior é um exemplo de argumento válido. Você pode ver que se ambas as premissas forem verdadeiras, a conclusão também terá que ser verdadeira.

Agora você pode ver por que afirmo, no tópico anterior, que a validade de um argumento está relacionada à sua estrutura. Quando há falha nessa estrutura, o argumento é inválido, ainda que todas as suas proposições sejam verdadeiras. Por exemplo, observe este argumento inválido:

Premissas:

> Pedro Álvares Cabral descobriu o Brasil.
>
> Albert Einstein propôs a Teoria da Relatividade.

Conclusão:

> Bill Gates é o homem mais rico do mundo.

Todas essas proposições são verdadeiras, mas isso não significa que o argumento seja válido. Nesse caso o argumento é inválido, pois não há uma estrutura que dê suporte à conclusão para que esta, obrigatoriamente, decorra dessas premissas. Se as ações da Microsoft de repente despencarem e Bill Gates perder toda a sua fortuna amanhã, as premissas serão verdadeiras e a conclusão falsa.

Estudando Exemplos de Argumentos

Já que os argumentos são tão importantes para a Lógica, incluirei neste tópico mais alguns exemplos para fazê-lo entender como eles funcionam.

Alguns desses exemplos possuem argumentos em que a primeira premissa está na forma “*Se...então...*” *Se* algo acontece, *então* algo mais acontecerá. Você pode pensar nesse tipo de frase como um escorregador: quando alguém se coloca na situação do *se* no topo do escorregador, acaba indo parar na situação do *então*, lá embaixo.

Aristóteles foi o primeiro a estudar as formas dos argumentos. Por exemplo:

Premissas:

Todos os homens são mortais.

Sócrates é homem.

Conclusão:

Sócrates é mortal.

Ele chamava essa forma de argumento de *silogismo* (veja o Capítulo 2 para aprender mais sobre as formas primárias da Lógica estudada por Aristóteles e outros pensadores).

Depois que você pegar o jeito de como os argumentos lógicos funcionam, as variações serão infinitas. Você descobrirá, da Parte II à Parte V deste livro, formas ainda mais precisas e úteis para criar, entender e provar argumentos lógicos. Por ora, esses exemplos apenas lhe darão uma amostra do que são exatamente os argumentos válidos.

Sorvete de domingo

Suponha que em um domingo qualquer, André, seu filho, lhe diga: “Você disse que se fôssemos ao parque domingo, poderíamos tomar sorvete. Se estamos indo ao parque, significa que vamos tomar sorvete.” A lógica dele está impecável. Para mostrar o porquê, eis o argumento de André, separado em premissas e conclusão:

Premissas:

Se formos ao parque, então poderemos tomar sorvete.

Nós estamos indo ao parque.

Conclusão:

Nós poderemos tomar sorvete.

A primeira premissa estabelece o escorregador “se – então” e a segunda mostra onde subimos nesse escorregador. Como resultado, inevitavelmente somos levados à conclusão.

O lamento de Fifi

Suponha que em uma tarde você chegue em casa da escola e sua mãe lhe apresente o seguinte argumento: “Se você se preocupasse com sua cadela Fifi, você a levaria para passear todo dia, depois da escola. Mas já que não faz isso, então não se preocupa com ela”. Eis o que você tem quando desmembra esse argumento em premissas e conclusão:

Premissas:

Se você se preocupasse com sua cadela Fifi, então você a levaria para passear todo dia, depois da escola.

Você não leva Fifi para passear todo dia, depois da escola.

Conclusão:

Você não se preocupa com sua cadela Fifi.

A primeira premissa estabelece o escorregador “se – então”. A segunda, no entanto, mostra que você *não* chega ao final do escorregador. A única maneira disso acontecer seria se você *não* subisse no escorregador. Assim, a conclusão de sua mãe é válida: pobre Fifi!

Fuga de Nova York

Imagine que sua amiga Ana, ao descrever onde mora, construa o seguinte argumento: “Manhattan é em Nova York. Hell’s Kitchen fica em Manhattan. Meu apartamento fica em Hell’s Kitchen e moro lá, então, moro em Nova York.” Esse argumento também está embasado em um escorregador “se – então”, embora não contenha as palavras “se” e “então”. Como elas estão implícitas no argumento, mas não aparecem, explico essa questão a seguir:

Premissas:

> Se alguma coisa está em meu apartamento, então ela está em Hell’s Kitchen.
>
> Se alguma coisa está em Hell’s Kitchen, então está em Manhattan.
>
> Se alguma coisa está em Manhattan, então está em Nova York.
>
> Eu moro no meu apartamento.

Conclusão:

> Eu moro em Nova York.

O escorregador “se – então” torna a conclusão evidente, nesse caso. Entretanto, nesse exemplo um escorregador leva a outro, que leva a outro. Depois que você sabe que Ana mora em seu apartamento, não tem outra opção, senão escorregar pelos próximos três escorregadores para chegar à conclusão de que ela mora em Nova York.

O caso da funcionária descontente

Suponha que sua mulher, Magda, chegue em casa do trabalho, zangada e diga: “Pode-se encontrar três tipos de chefes no mundo: aquele que lhe paga em dia, o que se desculpa quando lhe paga com atraso e aquele que não o valoriza como funcionário. Bem, meu salário está atrasado e meu chefe não se desculpou, então eu sei que ele não valoriza meu trabalho.”

Veja o argumento dela:

Premissas:

> Um chefe paga seus funcionários em dia ou se desculpa quando paga com atraso ou não valoriza o seu trabalho.
>
> Meu chefe não me pagou em dia.
>
> Meu chefe não se desculpou pelo atraso.

Conclusão:

> Meu chefe não me valoriza.

Esse argumento não se apoia em um escorregador “se – então”, mas em um conjunto de alternativas construídas com o uso da palavra “ou”. A primeira premissa estabelece as alternativas, enquanto a segunda e a terceira eliminam uma alternativa cada uma. A conclusão é a única alternativa que resta.

O que a Lógica Não É

Por existir há cerca de 2.000 anos, a Lógica teve a oportunidade de se entrelaçar na estrutura de quase tudo o que existe em nossa cultura. Dr. Spock, de *Jornada das Estrelas*, é apenas a ponta do iceberg.

Veja alguns dos estereótipos culturais que envolvem a Lógica: quando você conhece alguém calmo e atencioso, pensa que se trata de uma pessoa *lógica*. Quando alguém toma uma decisão impulsiva ou precipitada, ou quando se trata de alguém com quem não concorde, você prontamente o acusa de ser *ilógico* e o aconselha a *pensar logicamente* sobre o que está fazendo. Por outro lado, quando se depara com alguém frio ou imparcial, em geral você diz que é *uma* pessoa *guiada pela lógica.*

A palavra “lógica” é tão livremente empregada na linguagem informal que as pessoas acabam utilizando todo tipo de conceitos equivocados sobre ela. Em algumas áreas ela é reverenciada como a maior conquista da humanidade. Em outras, é menosprezada e encarada como um exercício para o isolamento, já que está muito distante de todos os desafios cotidianos que as pessoas enfrentam em suas vidas.

A Lógica acaba tendo essa reputação dúbia, de boa e de má, pois, na maioria das vezes, é vista como algo que não é.

Use a Tabela 3-1 como guia de referência para comparar o que a Lógica pode ou não fazer.

Tabela 3-1 O Que a Lógica Pode ou Não Fazer

A Lógica Não Pode	A Lógica Pode
Criar um argumento válido.	Criticar um determinado

	argumento quanto à validade.
Mostrar o que é verdadeiro ou falso na realidade.	Mostrar como trabalhar com proposições verdadeiras ou falsas.
Mostrar se um argumento é sólido.	Mostrar se um argumento é válido.
Justificar as conclusões encontradas pela indução.	Justificar as conclusões encontradas pela dedução.
Construir argumentos retoricamente mais fortes (mais convincentes).	Fornecer as bases para um aperfeiçoamento retórico.

Pensamento versus Lógica

O raciocínio ou mesmo um pensamento puro e simples é um processo imensamente complexo que só é compreendido superficialmente. Diversamente do que para o Dr. Spock e seu clã, a Lógica é apenas uma das peças do nosso processo de raciocínio, mas não é tudo.

Pense no tipo de raciocínio utilizado para resolver uma briga entre duas crianças. Você precisa observar o que está acontecendo, relacionar essa experiência com as anteriores e tentar descobrir o que pode acontecer no futuro. Pode optar entre agir severamente e ameaçá-las com uma punição, agir cordialmente e acalmá-las, permanecer imparcial e ouvir os dois lados da história ou usar um pouco de cada umas dessas abordagens.

Resumindo, você teria inúmeras opções para tentar buscar a paz (pode ser, também, que nenhuma dessas alternativas funcione, mas isso é um assunto para outro livro, e não para *Lógica Para Leigos*). Todas essas opções envolvem o pensamento. Ainda que sua decisão seja a de mandar cada criança para seu quarto sem tentar entender o que aconteceu, você estará usando o tipo de raciocínio que os humanos usam melhor (mas que nem mesmo o mais inteligente dos cachorros consegue usar).

No entanto, se você se limitar à Lógica para manter a paz entre as crianças, certamente não chegará a lugar algum. Ela não dirá o que fazer nem como agir, não é esse tipo de ferramenta. De modo que, quando reagir a uma situação observando, pensando e não deixando que as emoções controlem suas ações, perceba que estará invocando todo um conjunto de habilidades muito mais sofisticado (e também muito menos evidente!) do que a Lógica.

LEMBRE-SE

Não confunda Lógica com pensamento. Lógica é apenas um dos aspectos do pensamento, mas é um aspecto que vem sendo estudado de forma tão intensa que é muito bem compreendido. Nesse sentido, Lógica é muito mais — e também muito menos — que simplesmente pensar sem emoção.

Realidade — que conceito!

Na Lógica você está sempre trabalhando com proposições verdadeiras ou falsas, o que pode fazê-lo pensar que o ponto central da Lógica é mostrar o que é verdadeiro e o que é falso, em outras palavras, pode pensar que a intenção da Lógica é mostrar a natureza da realidade em termos objetivos. Mas, na verdade, não cabe à Lógica dizer o que é objetivamente verdadeiro ou falso uma vez que ela só pode mostrar isso em relação a outras proposições que você já sabe (ou acredita) serem verdadeiras ou falsas.

Por exemplo, veja estas duas proposições:

> Nova Jérsei é um estado dos Estados Unidos da América.
>
> Roubar é sempre errado.

Parece simples confirmar a primeira proposição como verdadeira e difícil de analisar a segunda. Porém, em ambos os casos, a Lógica não pode distinguir o que é certo do que é errado, e nem mesmo dirá se Nova Jérsei é mesmo um estado — você terá que encontrar outras fontes para confirmar essa informação. O que a Lógica *pode fazer* é dizer se um argumento é válido, isto é, se um conjunto de premissas verdadeiras produz uma conclusão verdadeira.

A única exceção a essa regra envolve proposições que são verdadeiras apenas em um sentido muito limitado e banal. Por exemplo:

Todas as coisas azuis são azuis.

Tom é um cachorro ou não é um cachorro.

Se você é um vendedor que mora em Denver, então você é um vendedor.

Proposições como essas são chamadas de *tautologias e* têm um papel importante na Lógica, como você verá no Capítulo 6. Por ora, apenas é necessário saber que o principal motivo para as tautologias serem consideradas verdades tão incontestáveis é porque não transmitem qualquer informação a respeito do mundo. Para descobrir o que é verdadeiro ou falso nesse complexo e louco mundo você está por conta própria.

A solidez da coerência

Um *argumento sólido ou coerente* é simplesmente um argumento válido com premissas verdadeiras que fazem com que a conclusão seja, também, verdadeira.

A solidez de um argumento anda de mãos dadas com a realidade. Mas, tendo em vista que a Lógica não é capaz de dizer se uma proposição é verdadeira ou falsa, também não é capaz de dizer se um argumento é sólido. Mesmo um argumento válido pode produzir uma conclusão ruim se partir de premissas falhas.

Premissas:

> Jennifer Lopez está na Terra.
>
> Quando o Messias estiver na Terra, o final dos tempos terá chegado.
>
> Jennifer Lopez é o Messias.

Conclusão.

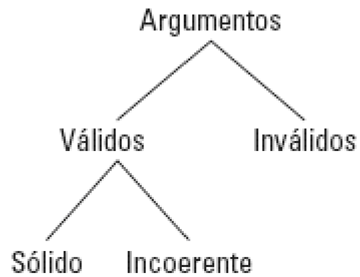
> O final dos tempos chegou.

No exemplo acima tem-se um argumento logicamente válido. No entanto, ele é sólido? Pessoalmente, não consegui aceitar a terceira premissa como sendo verdadeira, então teria que afirmar que esse argumento não é sólido; você pode concordar comigo. O ponto é: se a Lógica não decide se essa proposição é verdadeira na vida real, ela não pode ajudar a decidir se o argumento é sólido.

Certifique-se de conseguir diferenciar um argumento válido daquele que é sólido ou coerente. O válido contém um grande se: *se* todas as premissas são verdadeiras, a conclusão também tem que ser verdadeira. O sólido ou coerente é um argumento válido com uma condição adicional: as premissas *realmente são verdadeiras*, então, é claro que a conclusão também é verdadeira.

Observe que quando você parte de um argumento inválido, a Lógica tem muito a dizer: se o argumento é inválido, também é frágil ou incoerente. A Figura 3-1 mostra uma estrutura em forma de árvore que pode ajudar a entender melhor os argumentos.

Figura 3-1: Observar a árvore lógica pode ajudar a classificar seus argumentos.



Dedução e indução

DICA

Dedução rima com *redução*, o que facilita na hora de lembrar que na dedução você parte de um conjunto de possibilidades e o *reduz* até que reste um subconjunto menor.

As histórias de mistério, por exemplo, são um ótimo exercício de dedução. Normalmente o detetive começa com um conjunto de possíveis suspeitos tais como: o mordomo, a arrumadeira, o sócio e a viúva. Ao final da história ele terá reduzido esse conjunto a apenas uma pessoa, como por exemplo: “A vítima morreu na banheira, mas foi removida até a cama. Porém, nenhuma mulher conseguiria levantar o corpo, nem mesmo o mordomo, pois ele tem um ferimento de guerra. Portanto, o sócio deve ter cometido o crime.,”

DICA

Indução começa com as mesmas duas letras que a palavra *incremento*, o que ajuda a lembrar que na indução você começa com um número limitado de observações e o *incrementa* por generalização.

Por exemplo, suponha que você passe o final de semana em uma pequena cidade e as primeiras cinco pessoas que conheça sejam gentis. Indutivamente, você conclui o seguinte: “Todos nesta cidade são tão gentis”. Em outras palavras, começou com um pequeno conjunto de exemplos e incrementou esse conjunto para incluir um maior.

A Lógica lhe permite racionalizar de forma dedutiva e com confiança. Na verdade, ela é feita sob medida para que se possa separar um grupo de proposições fáticas (*premissas*), descartando as proposições plausíveis, mas imprecisas (*conclusões inválidas*), e chegando à verdade (*conclusões válidas*). Essa é a razão para a Lógica e a dedução estarem intimamente ligadas.

A dedução funciona especialmente bem na Matemática, onde os objetos de estudo são claramente definidos e existem poucas ou nenhuma zona cinzenta. Por exemplo, cada um dos números cardinais é par ou ímpar. Assim, se quiser provar que um número é ímpar, basta refutar a hipótese de ele ser dividido por 2. Veja o tópico “De Quem É Esta Lógica, Afinal de Contas?” para saber mais sobre como e onde a Lógica é aplicada.

Por outro lado, embora a indução pareça ser muito útil, ela é logicamente falha. Conhecer 5 pessoas simpáticas — ou 10 ou 10.000 — não é garantia de que a próxima que irá conhecer não seja chata. Conhecer 10.000 pessoas não garante nem mesmo que a maioria das pessoas da cidade sejam gentis pois você pode ter conhecido apenas todas as gentis.

Lógica, no entanto, é mais do que um bom palpite de que uma conclusão esteja correta. A definição de validade da Lógica requer que caso as premissas sejam verdadeiras, a conclusão também o seja. A indução, por não ter essa característica, é considerada o grande elefante branco, tanto da Ciência quanto da Filosofia: parece útil, mas só ocupa um enorme espaço na sala.

Perguntas retóricas

Retórica é o estudo daquilo que torna um argumento cogente e convincente.

Observe o seguinte argumento:

Premissas:

A ciência não pode explicar tudo na natureza.

Qualquer coisa na natureza pode ser explicada pela ciência ou pela existência de Deus.

Conclusão:

Deus existe.

Mesmo que esse argumento possa ser sólido ou não, ele é válido. No entanto, pode não ser *cogente*.

O que torna um argumento cogente é muito parecido com aquilo que o torna sólido (veja o tópico “A solidez da coerência” já tratado neste capítulo). Em um argumento sólido, as premissas são indubitavelmente verdadeiras. No cogente, as premissas são verdadeiras, além de mostrarem-se plausíveis.

A palavra “plausível” pode lembrá-lo do jargão dos tribunais. Essa conexão faz sentido porque advogados constroem seus argumentos lógicos com a intenção de convencer o juiz ou os jurados de uma determinada conclusão. A fim de serem cogentes, os argumentos dos advogados devem ser válidos e suas premissas precisam ser verossímeis (veja o tópico “Diga isso ao juiz (direito)”, onde discutirei a relevância da Lógica no contexto legal).

A ciência da indução

Indução é o raciocínio que parte de um número limitado de observações em direção a uma conclusão geral. Um exemplo clássico é: depois de observarmos que 2 ou 10 ou 1.000 corvos são pretos, concluímos que todos os corvos são pretos.

Embora possam ser convincentes, argumentos indutivos não são logicamente válidos (você não sabe se existe por aí um corvo branco ou cor de rosa). Mas, embora a indução seja logicamente falível, os cientistas parecem usá-la todo o tempo, com enorme sucesso para explicar todos os tipos de fatos do universo. O filósofo Karl Popper usou essa aparente contradição para demonstrar que a indução não é necessária para a descoberta científica.

Em poucas palavras, a resposta de Popper é que a ciência não usa a indução para provar nada — é apenas aparência. Ao invés disso, Popper diz que os cientistas desenvolvem teorias que se encaixam em suas observações e então tentam *refutar* as teorias alternativas. Então, a teoria não refutada passa a ser aceita como a explicação para aquela teoria até que surja outra melhor. Isso pode parecer fácil, como se qualquer mentecapto pudesse inventar uma teoria maluca e fazê-la ser aceita; mas os cientistas são peritos em refutar teorias plausíveis, então ainda não largue seu emprego.

Mesmo que alguns pensadores ainda questionem a explicação de Popper, muitos acham que é uma explicação válida, que resolve um problema filosófico centenário.

Se você considerar que a Lógica diz respeito à validade e não à solidez, qual o seu papel no processo de tornar um argumento convincente? Simples, criando uma clara distinção entre o que é forma básica de um argumento e o que é seu conteúdo.

A forma do argumento é o que diz respeito à Lógica. Se a forma não funcionar corretamente, seu

LEMBRE-SE

argumento desmorona. Entretanto, quando ele funciona corretamente, você pode prosseguir com confiança para a análise do seu conteúdo.

Volte um pouco e observe o argumento do início deste tópico. Foi dito que é um argumento válido, mas que pode não ser cogente. Por quê? Pelo simples fato de a segunda premissa estar cheia de falhas. Considere o seguinte:

- ✓ Se a ciência um dia puder explicar tudo sobre a natureza, o que hoje não faz?
- ✓ O que acontece se algumas coisas puderem ser explicadas por algo que não seja a ciência ou a existência de Deus?
- ✓ Se simplesmente não houver explicação para tudo o que vemos na natureza?

Cada uma dessas perguntas é uma exceção plausível para o argumento, e você precisa lidar com elas se quiser convencer pessoas inteligentes e racionais de que seu raciocínio está correto. Em outras palavras, precisa lidar com as perguntas retóricas de seu argumento.

Entretanto, na realidade, apelar para a inteligência da plateia pode ser menos eficiente do que apelar para seus sentimentos mais profundos ou suas crenças irracionais. Em alguns casos, um argumento simples e frágil pode ser mais convincente do que um sólido que seja de difícil compreensão por parte da plateia. Um argumento trôpego de um orador de quem a plateia goste pode ser convincente, enquanto que uma apresentação brilhante de um orador pernóstico pode ser desinteressante.

O estudo sobre o que torna um argumento cogente ou convincente é muito útil e instigante, mas está fora do escopo deste livro. De um ponto de vista puramente lógico, uma vez que o argumento é válido, não pode ser melhorado.

De Quem É Esta Lógica, Afinal de Contas?

Com todas essas restrições que lhe são impostas, você pode pensar que a Lógica é muito limitada para que possa ter alguma utilidade. Mas essa limitação é, na realidade, sua maior força. A Lógica é como um laser, uma ferramenta mais bem usada para dar foco do que para iluminar. O laser pode não iluminar a sua casa, mas, como a Lógica, seu grande poder está na precisão. Os tópicos a seguir descrevem apenas algumas das áreas onde ela é mais utilizada.

Escolha um número (Matemática)

A Matemática é feita sob medida para o uso da Lógica em todo o seu potencial. Na verdade a Lógica é um dos três maiores alicerces teóricos sobre os quais a Matemática está construída, os outros dois são a Teoria dos Conjuntos e a Teoria dos Números, caso você esteja se perguntando.

A Lógica e a Matemática funcionam muito bem juntas, pois ambas independem da realidade e são ferramentas utilizadas para ajudar as pessoas na tarefa de dar um sentido ao mundo. Por exemplo, a realidade pode conter três maçãs ou quatro bananas, mas as ideias de *três* e *quatro* são abstrações, ainda que as pessoas, em sua maioria, as tomem como verdades absolutas.

A Matemática é totalmente construída sobre abstrações. Quando essas abstrações ficam mais complicadas — como na Álgebra, no Cálculo e assim por diante —, a Lógica pode ser invocada para ajudar a trazer uma certa ordem para essa complexidade. As ideias matemáticas, assim como os números, a soma, a fração e outras, são claramente definidas, sem exceções. É por isso que as proposições sobre essas ideias são muito mais fáceis de serem confirmadas do que uma que trate da realidade, tal como “As pessoas, em geral, têm bom coração”, ou mesmo “Todos os corvos são pretos”.

Leve-me para a Lua (Ciência)

A ciência usa a Lógica para obter uma enorme vantagem. Assim como a Matemática, a ciência usa abstrações para fazer com que a realidade faça sentido e aí aplica a Lógica nessas abstrações.

A ciência tenta entender a realidade da seguinte forma:

1. Reduz a realidade a um conjunto de abstrações, chamado de *modelo.*
2. Trabalha com esse modelo para chegar a uma conclusão.
3. Aplica essa conclusão na realidade.

A Lógica do segundo passo é instrumental e as conclusões a que a ciência chega são, obviamente, conclusões lógicas. Esse processo é mais eficaz quando existe uma boa correlação entre o modelo e a realidade, e quando o modelo é adequado para os tipos de cálculos com que a Lógica trabalha melhor.

As áreas da ciência que mais se apoiam na Lógica e na Matemática são as *Ciências Quantitativas*, tais como Física, Engenharia e Química. As *Ciências Qualitativas* — Biologia, Fisiologia e Medicina — usam a Lógica, mas com um pouco menos de convicção. Finalmente, as *Ciências Sociais*, tais como Psicologia, Sociologia e Economia, são as ciências onde os modelos trazem pelo menos uma direta correlação com a realidade, o que significa que elas tendem a se apoiar menos na Lógica pura.

Ligue e desligue (Ciência da Computação)

A Medicina costumava ser chamada de ciência mais jovem, mas, agora, esse título pertence à Computação. Grande parte do sucesso da revolução do computador reside na Lógica.

Toda ação realizada pelo computador acontece por causa de uma complexa estrutura de instruções lógicas. No hardware — a estrutura física da máquina —, a Lógica é instrumental no projeto dos complexos circuitos que tornam o computador possível. Já no software — os programas que tornam o computador útil —, a linguagem do computador que é baseada na Lógica fornece a este a infinita versatilidade que o distingue de todas as outras máquinas.

Veja o Capítulo 20 para uma discussão mais aprofundada sobre a Lógica no que diz respeito aos computadores.

Diga isso ao juiz (Direito)

Como acontece com a Matemática, o direito existe primariamente como um conjunto de definições: *contratos, delitos, crimes, dolo* e assim por diante. Todos esses conceitos são descritos e depois aplicados a casos específicos e interpretados pelos tribunais. Uma definição legal fornece as bases para o argumento legal, que é semelhante ao lógico.

Por exemplo, para demonstrar uma infração a direitos autorais, o requerente pode precisar provar que o réu publicou uma determinada quantidade de material como sendo de sua autoria, em troca de compensação, monetária ou não, quando esse material era protegido por direitos autorais preexistentes.

Esses critérios são semelhantes às premissas de um argumento lógico: se eles forem considerados verdadeiros, a conclusão — de que o réu cometeu a infração a direitos autorais — também tem que ser verdadeira.

Encontre o sentido da vida (Filosofia)

A Lógica nasceu na Filosofia e, em geral, ainda é ensinada como um dos ramos da Filosofia e não da Matemática. Aristóteles inventou a Lógica como um método para compreender a estrutura básica da razão, que ele viu como o motor propulsor da tentativa da humanidade entender o universo nos seus sentidos mais amplos.

LEMBRE-SE

Assim como a Ciência, a Filosofia se apoia em modelos de realidade para ajudar a explicar tudo aquilo que vemos. Entretanto, tendo em vista que os modelos raramente são matemáticos, os filósofos tendem a se inclinar mais na direção da Lógica Retórica do que na da Matemática Lógica.

Parte II:

Lógica Sentencial Formal (LS)

Nesta parte...

Se você já deu uma olhadinha em um livro de Lógica (talvez naquele que está esquecido na sua mesa, acumulando poeira desde o início do semestre!), já deve ter imaginado o que significam todos aqueles símbolos esquisitos → e ↔; e aqui é onde vai descobrir. Esta parte será exclusivamente sobre a Lógica Sentencial ou LS, onde todos esses símbolos entram em jogo.

No Capítulo 4, você descobrirá como traduzir proposições do português para a LS usando as constantes, as variáveis e os cinco operadores lógicos. O Capítulo 5 traz dicas para quando você entrar no traiçoeiro mundo da valoração das proposições na LS, a fim de decidir se são verdadeiras ou falsas. O Capítulo 6 traz as tabelas verdade, que são poderosos instrumentos para desvendar uma série de coisas sobre as proposições na LS. O Capítulo 7 mostra como as tabelas rápidas substituem as tabelas verdade como um meio mais rápido de resolver problemas. Finalmente, o Capítulo 8 traz as árvores lógicas, um instrumento que reúne todas as vantagens das tabelas verdade e das tabelas rápidas, mas nenhuma de suas desvantagens.

Capítulo 4
Assuntos Formais

Neste Capítulo

- Apresentando a Lógica Formal
- Definindo os cinco operadores lógicos
- Traduzindo proposições

Se você olhar para alguns dos argumentos lógicos contidos no Capítulo 3, poderá ter a impressão de que todos eles têm muito em comum, o que é verdade. Ao longo dos séculos, os lógicos examinaram uma grande quantidade de exemplos de argumentos e observaram que alguns padrões se repetem. Esses padrões podem ser representados por um pequeno número de símbolos que podem, depois, ser estudados por suas características comuns.

Neste capítulo, eu o apresento à *Lógica Formal*, que é um conjunto de métodos infalíveis para determinar se um argumento é válido ou inválido. Mostro como representar proposições com símbolos, chamados de *constantes* e *variáveis*, e você conhecerá os cinco *operadores lógicos* usados para relacionar proposições simples a outras mais complexas.

Os operadores lógicos funcionam de forma muito semelhante aos aritméticos (como a adição, subtração e assim por diante), o que me leva a mostrar as semelhanças entre eles para que você possa se familiarizar com esses novos símbolos. Finalmente, você verá como traduzir as proposições do português para a Lógica e vice-versa.

Observando as Formalidades da Lógica Sentencial

Conforme apreciado no Capítulo 2, a *Lógica Sentencial* (ou LS, também conhecida como *Proposicional*) é uma das duas formas da Lógica Formal Clássica (a outra é a *Quantitativa*, ou LQ, também conhecida como *Predicativa)*. Apresento a LS neste capítulo e a examinarei em detalhes nas Partes II e III. A LQ será estudada na Parte IV.

Os argumentos lógicos são construídos sob a forma de linguagem, mas as linguagens naturais, tais como o português, tendem a ser um pouco ambíguas. As palavras geralmente têm mais de um significado e as frases podem ser mal interpretadas.

Para ajudar a resolver esse problema, matemáticos e filósofos desenvolveram a Lógica Sentencial, uma linguagem especialmente criada para expressar argumentos lógicos com precisão e clareza. Por ser simbólica, a LS possui a vantagem adicional de permitir o cálculo de acordo com regras e fórmulas precisamente determinadas. Tal como acontece na Matemática, para garantir a resposta certa basta seguir as regras corretamente.

Nos tópicos seguintes apresento alguns tipos de símbolos que a LS usa para atingir esses objetivos.

Constantes da proposição

Se alguma vez você já passou o dia assistindo a uma aula de Álgebra ou Préálgebra, certamente já foi apresentado a esse enigmático companheiro conhecido como *x*. Provavelmente já lhe falaram que o *x* substitui um número secreto e que o seu trabalho é fazer com que esse *x* fale. O(a) professor(a) deve ter demonstrado todos os tipo de formas sádicas para se torturar o pobre *x* até que ele ceda e revele sua verdadeira identidade numérica. Como deve ter sido divertido!

A substituição de números por letras é uma das coisas em que os matemáticos são realmente bons. Assim, não é de surpreender que a

Lógica Formal, que foi desenvolvida por matemáticos, também use letras como substitutas. Na introdução deste capítulo, deixei claro que a Lógica usa *proposições* ao invés de números, de forma que é racional supor que na Lógica Formal as letras substituem proposições. Por exemplo:

> Considere *K* = Kátia está alimentando seu peixe.
>
> Considere *F* = Os peixes estão sacudindo suas pequenas nadadeiras alegremente.

Quando uma letra substitui uma proposição em português, ela é chamada de *constante da proposição.* Por convenção, são usadas letras maiúsculas para as constantes.

Tratando-se de constantes, os lógicos gostam mais das letras *P* e *Q*. Alguns colegas dizem que é porque *P* é a primeira letra da palavra *proposição* e *Q* porque é a letra seguinte no alfabeto. Minha teoria é a de que depois de estudar toda aquela Álgebra na escola, eles simplesmente se cansaram de usar *X* e *Y.*

Variáveis da proposição

Quando os lógicos entenderam a ideia da substituição das proposições por letras nunca mais pararam. Eles perceberam que poderiam usar uma letra para *qualquer* proposição, até mesmo para uma proposição da LS. Quando as letras são usadas dessa maneira, são chamadas de *variáveis da proposição*.

Ao longo deste livro usarei as variáveis para mostrar os padrões gerais da LS e as constantes para os exemplos mais detalhados.

As letras que substituem uma proposição da LS são chamadas de *variáveis da proposição.* Por convenção, letras minúsculas são usadas para as variáveis. Neste livro, usarei *x* e *y* na maioria dos exemplos, e quando necessário usarei também *w* e *z.*

Valor lógico

Conforme já esclareci no Capítulo 3, toda proposição tem um *valor lógico* que pode ser verdadeiro ou falso. Na Lógica Formal, *verdadeiro* é abreviado para **V** e *falso* para **F.**

Como exemplo, considere os valores lógicos destas duas constantes da proposição:

Considere *N* = O Nilo é o rio mais longo da África.

Considere *L* = Leonardo DiCaprio é o rei do mundo.

Aqui é verdade que o Nilo é o rio mais longo da África, então o valor verdade de *N* é **V**. Mas, Leonardo DiCaprio *não* é o rei do mundo, então o valor lógico de *L* é **F.**

A *Álgebra Booleana*, precursora da Lógica Formal, usa o valor *1* para representar **V** e *0* para representar **F**. Esses dois valores ainda são utilizados na Lógica Computacional (a Álgebra Booleana é examinada no Capítulo 14 e a Lógica Computacional no Capítulo 20).

Os Cinco Operadores da LS

A LS tem cinco operadores básicos, como você pode ver na Tabela 4-1. Esses *operadores lógicos* são semelhantes aos operadores aritméticos, na medida em que assumem valores que lhes são atribuídos para produzir um novo valor. Entretanto, os operadores lógicos só lidam com dois valores: os valores lógicos, **V** e **F**. Nos tópicos seguintes, explico cada um dos operadores contidos na Tabela 4-1.

Tabela 4-1: Os Cinco Operadores Lógicos

Operador	Nome Técnico	Significado	***Exemplo***
~	Negação	Não	~x
^	Conjunção	E	x ^ y
v	Disjunção	Ou	x v y
→	Condicional	Se...então	x → y
↔	Bicondicional	Se e somente se	x ↔ y

Sentimento negativo

Você pode transformar qualquer proposição em seu oposto apenas adicionando ou trocando algumas palavras. Isso é chamado de *negação* da proposição. É óbvio que ao negar uma proposição verdadeira, ela se transforma em falsa e, ao negar uma falsa, esta se transforma em verdadeira. De modo geral, toda proposição tem um valor lógico contrário ao da sua negação.

Por exemplo, posso adaptar a proposição *N* abaixo simplesmente inserindo uma pequena palavra:

N = O Nilo é o rio mais longo da África.

~*N* = O Nilo *não* é o rio mais longo da África.

A adição da palavra *não* transforma a proposição original em seu oposto, que é sua negação. Tendo estabelecido que o valor de *N* é **V** (veja o tópico “Valor lógico” no início deste capítulo), podemos concluir que o valor lógico de sua negação é **F.**

Na LS o operador de negação é o *til* (~). Veja outro exemplo de negação:

L = Leonardo DiCaprio é o rei do mundo.

~*L* = Leonardo DiCaprio *não* é o rei do mundo.

Nesse caso, tendo estabelecido que o valor de *L* é **F** (veja o tópico “Valor lógico” no início deste capítulo), você também pode concluir que o valor d*ê ~L* é **V.**

Essa informação é facilmente resumida na tabela a seguir:

x	~x
V	F
F	V

Memorize a informação contida nessa tabela, pois a usará diversas vezes nos demais capítulos.

Como pode ver na tabela I, a variável *x* foi usada para substituir qualquer proposição de LS. Quando essa proposição, representada pelo *x*, é verdadeira, então aquela representada por ~*x* é falsa. Por outro lado, quando a proposição representada por *x* é falsa, então a~*x* é verdadeira.

Alguns livros de Lógica usam o hífen (–), ou outro símbolo parecido com um L deitado para o operador *não* ao invés do til. Prefiro o til, mas qualquer que seja o símbolo, o significado será o mesmo.

DICA

Sentimento um pouco mais negativo

Embora a negação seja apenas o começo, o pequeno sistema de símbolos da LS é muito mais poderoso do que parece. Por exemplo, considerando-se que o valor de uma nova proposição ~*R* é **V** e que sua negaçã*ôR* é **F**, o que você pode dizer a respeito de ~~*R*?

Se você imaginou que o valor d*ê* ~~*R* é **V**, parabéns! Perceba que é capaz de afirmar isso sem nem mesmo conhecer a proposição *R*.

Testemunhe a mágica e o poder da Lógica! Com apenas algumas regras simples, você sabe que *qualquer* proposição desse formato será *obrigatoriamente* verdadeira, mesmo que não saiba exatamente qual é a proposição. Essa garantia tem como fundamento a mesma noção de que basta saber que 2 maçãs + 3 maçãs = 5 maçãs para que saiba que esse resultado será verdadeiro, não importa o que esteja somando: maçãs, dinossauros, duendes – o que quer que seja.

Tabelando o movimento

Tendo em vista que *R* pode ter apenas dois valores lógicos **V** ou **F**, você pode organizar as informações sobre ~*R*, ~~*R* e assim por diante na tabela:

R	~R	~~R	~~~R
V	F	V	F
F	V	F	V

As tabelas desse tipo são chamadas de *tabela verdade*. A primeira coluna contém os dois valores lógicos possíveis de *R*: **V** e **F**. As demais colunas fornecem os

valores correspondentes para as diferentes proposições relacionadas: $\sim R$, $\sim\sim R$ e assim por diante.

Interpretar tabelas verdade é muito fácil. Usando a amostra acima, se você sabe que o valor de R é **F** e quer saber o valor de $\sim\sim\sim R$, basta encontrar o ponto em que a última linha encontra a última coluna. O valor lógico nessa posição diz que quando R é **F**, $\sim\sim\sim R$ é **V**.

O Capítulo 6 mostra que as tabelas verdade são um poderoso instrumento da Lógica, mas, por enquanto, vou utilizá-las apenas para organizar as informações de modo mais claro.

Apresentando os "es"

O símbolo ^ é chamado de *operador de conjunção* ou simplesmente *operador ^. Você* pode pensar nele simplesmente como a palavra *e* sendo colocada entre duas proposições, juntando-as para formar uma nova.

Veja esta proposição:

> Porto Alegre é a capital do Rio Grande do Sul *e* Pelé foi jogador do Santos F.C..

Essa proposição é verdadeira ou falsa? Para decidir você precisa reconhecer que ela contém duas proposições menores: uma sobre Porto Alegre e outra sobre Pelé. Seu valor lógico depende do valor de ambas as partes.

Tendo em vista que as duas são verdadeiras, a proposição como um todo é verdadeira. Suponha, no entanto, que uma delas seja falsa. Imagine algum universo alternativo onde Porto Alegre não seja a capital do Rio Grande do Sul, ou em que Pelé nunca tenha sido jogador do Santos. Em qualquer dos casos o valor lógico da proposição seria falso.

Na Lógica você lida com as proposições que envolvem a palavra *e* de um modo especial. Primeiro uma constante é atribuída para cada uma das proposições menores:

Considere *A* = Porto Alegre é a capital do Rio Grande do Sul.

Considere *J* = Pelé foi jogador do Santos F.C.

Então relacione as duas constantes da seguinte forma:

A ^ *J*

O valor lógico para essa nova proposição é baseado no valor lógico das duas partes que foram interligadas. Se essas partes forem *ambas* verdadeiras, então a proposição inteira será verdadeira. Por outro lado, se *qualquer uma* das partes (ou ambas) for falsa, então a proposição inteira será falsa.

Para o operador ^, você pode organizar os valores verdades de *x* e de *y* na tabela:

x	y	x^y
V	V	V
V	F	F
F	V	F
F	F	F

Memorize a informação contida nessa tabela. Veja um modo rápido de gravála: uma proposição ^ só é verdadeira quando ambas as partes que a formam são verdadeiras. Caso contrário ela é falsa.

Observe que a tabela para o operador ^ acima tem quatro linhas ao invés de apenas duas, como acontece com o operador ~ (veja o tópico “Tabelando o movimento” no início deste capítulo). As tabelas são diferentes porque o operador $\wedge$ sempre opera com duas variáveis, de forma que a tabela dele tem que abranger todos os quatro pares de valores para *x* e *y.*

DICA

Outros livros de Lógica podem utilizar um ponto (⊂) ou um & para o operador $\wedge$, ao invés do “e comercial”. Em outros livros, $x \wedge y$ é simplesmente escrito como *xy*. Seja qual for a convenção, o significado será o mesmo.

Cavando em busca do “ou”

Assim como ocorre com o *e*, uma proposição pode ser composta por duas menores unidas pela palavra *ou*. A Lógica possui um operador para a palavra *ou:* o *de disjunção*, ou simplesmente o *operador ou*, que é representado pelo símbolo $\vee$.

Veja a seguinte proposição:

> Porto Alegre é a capital do Rio Grande do Sul *ou* Pelé foi jogador do Santos F.C.

Se você designar a primeira parte da proposição de *A* e a segunda de *Q*, poderá juntar as duas constantes *A* e *Q* como segue:

> $A \vee Q$

Essa proposição é verdadeira? Assim como acontece com a $\wedge$, quando as duas partes de uma proposição $\vee$ são verdadeiras, a proposição inteira é verdadeira. Portanto, $A \vee Q$ tem um valor lógico **V**. Entretanto, em uma proposição $\vee$, mesmo se apenas uma das partes for verdadeira, a proposição como um todo ainda será verdadeira. Por exemplo:

> Considere *A* = Porto Alegre é a capital do Rio Grande do Sul.
>
> Considere *S* = Pelé foi jogador do São Paulo F.C.

Agora veja que a proposição $A \vee S$ significa:

> Porto Alegre é a capital do Rio Grande do Sul *ou* Pelé foi jogador do São Paulo F.C.

Muito embora a segunda parte seja falsa, a proposição inteira é verdadeira porque uma de suas partes é verdadeira. Portanto, *A* ᵛ *S* tem um valor lógico **V.**

Mas quando *ambas* as partes são falsas, a proposição ᵛ inteira é falsa. Por exemplo:

> Considere *Z* = Porto Alegre é a capital da Nova Zelândia.
>
> Considere *S* = Pelé foi jogador do São Paulo F.C.

Agora a proposição *Z* ᵛ *S* significa:

> Porto Alegre é a capital da Nova Zelândia *ou* Pelé foi jogador do São Paulo F.C.

Essa é uma proposição falsa porque suas duas partes são falsas. Então *Z* ᵛ *S* tem o valor de **F**.

Para o operador ᵛ, você pode construir a tabela com quatro linhas que abrangem cada uma das quatro possíveis combinações dos valores verdade de *x* e *y*:

x	y	x^y
V	V	V
V	F	V
F	V	V
F	F	F

Memorize a informação nessa tabela. Um modo rápido de guardar essa informação é: uma *proposição ou* só é falsa quando ambas as suas partes são falsas. Caso contrário ela é verdadeira.

Em português a palavra *ou* tem dois significados distintos:

- ✓ ***Ou* inclusivo:** Quando o *ou* significa "esta escolha *ou* aquela escolha, *ou ambas";* a possibilidade de que ambas as partes da proposição sejam verdadeiras está *incluída*. Um exemplo de um *ou* inclusivo é quando uma mãe diz: "Antes de sair, você precisa arrumar seu quarto *ou* fazer a lição de casa". Obviamente, ela quer dizer para o filho fazer uma das duas tarefas *ou ambas.*
- ✓ ***Ou* exclusivo:** Quando o *ou* significa "esta escolha *ou* aquela escolha, *mas não ambas*", a possibilidade de ambas as partes da proposição serem verdadeiras é *excluída*. Um exemplo de um *ou* exclusivo é quando uma mãe diz: "Eu vou lhe dar dinheiro para ir ao shopping hoje *ou* para o passeio a cavalo amanhã". Ela quer dizer que o filho terá o dinheiro apenas para um dos passeios, *mas não para ambos.*

A língua portuguesa é ambígua, mas a Lógica não. Por convenção, na Lógica o operador ∨ é sempre *inclusivo*. Se ambas as partes de uma proposição ∨ forem verdadeiras, ela inteira também será verdadeira.

Tanto o *ou* inclusivo quanto o exclusivo são usados em projetos de portas lógicas, uma das partes integrantes do *hardware* do computador. Veja o Capítulo 20 para saber mais sobre a Lógica Computacional.

Ficando em dúvida

O símbolo → é chamado de *operador condicional*, também conhecido por operador *se...então...* ou, apenas, operador *se.* Para entender como o operador → funciona, observe a seguinte proposição:

Se há uma peruca pendurada na cabeceira da cama do quarto de hóspedes, então tia Dóris está nos visitando.

Observe que a proposição contém duas proposições separadas, podendo cada uma delas ser representada pela seguinte constante:

Considere *W* = Uma peruca está pendurada na cabeceira da cama do quarto de hóspedes.

Considere *D* = Tia Dóris está nos visitando.

Relacione então as duas proposições com este novo operador:

$W \rightarrow D$

Assim como ocorre com os outros operadores tratados neste capítulo, você pode construir para o operador → uma tabela com quatro linhas que abrangem todas as possíveis combinações de valores verdade para *x* e *y*:

x	y	x → y
V	V	V
V	F	F
F	V	V
F	F	V

Memorize a informação contida nessa tabela. Uma maneira rápida de fazer isso é: uma proposição se só é falsa quando sua primeira parte é verdadeira e a segunda falsa. Caso contrário ela é verdadeira.

Outros livros de Lógica podem usar o símbolo □ para o operador se, ao invés da seta. Qualquer que seja o símbolo, o significado será o mesmo.

A aparência do operador → não é aleatória. A seta aponta da esquerda para a direita por um motivo importante: quando uma proposição se é verdadeira e sua primeira parte também o é, a segunda *tem que ser igualmente verdadeira*.

Preciso de duas novas proposições para tornar isso mais claro:

Considere *B* = Você está em Cuiabá.

Considere *M* = Você está no Mato Grosso.

Agora considere a proposição

$B \rightarrow M$

Essa proposição significa: “se você está em Cuiabá, então você está no Mato Grosso”. Obviamente a proposição é verdadeira, mas por quê? Porque Cuiabá está dentro do Mato Grosso.

A conversão de uma proposição

Quando você inverte uma proposição se, o resultado é uma nova proposição chamada de *conversão; eis um* exemplo:

Proposição se: Se você está em Cuiabá, então você está no Mato Grosso.

Conversão: Se você está no Mato Grosso, então você está em Cuiabá.

O fato de uma proposição se ser verdadeira não significa necessariamente que sua *conversão* também o seja. Apesar da proposição original acima ser verdadeira, sua conversão é falsa. Você poderia estar em Cáceres, Campo Verde ou em qualquer outro lugar do Mato Grosso.

A inversão de uma proposição

Quando você nega ambas as partes de uma proposição se, chega a uma outra proposição, chamada de *inversão.* Veja o exemplo:

> **Proposição se:** Se você está em Cuiabá, então está no Mato Grosso.
>
> **Inversão:** Se você não está em Cuiabá, então não está no Mato Grosso.

Quando uma proposição se é verdadeira, isso não quer dizer necessariamente que sua *inversão* também o seja. Usando o exemplo anterior, mesmo que você não esteja em Cuiabá, pode estar em outro lugar do Mato Grosso.

A contraposição de uma proposição

Quando você faz as *duas coisas*, isto é, inverte a ordem *e* nega as duas partes de uma *proposição–se*, obtém sua *contraposição.* Já sei, está ficando mais complicado! Mas veja o exemplo:

> **Proposição se:** Se você está em Cuiabá, então está no Mato Grosso.
>
> **Contraposição:** Se você não está no Mato Grosso, então não está em Cuiabá.

Uma *proposição-se* e sua contraposição sempre têm o mesmo valor lógico. Usando o exemplo anterior, se a primeira parte é verdadeira — você não está no Mato Grosso —, é claro que não pode estar em Cuiabá.

Embora uma proposição e sua contraposição sempre tenham o mesmo valor verdade, na prática, a contraposição é, às vezes, mais fácil de ser provada. (Para saber mais sobre provas na LS, dê uma olhadinha na Parte III.) A conversão de uma proposição tem sempre o mesmo valor verdade que a inversão dessa mesma proposição. Isso se deve ao fato de que a conversão e a inversão são, na verdade, contraposições *uma da outra*.

Ficando ainda mais em dúvida

Na LS o operador se...e...somente...se (↔) é parecido com o *se* (veja o tópico “Ficando em dúvida” neste capítulo). A melhor maneira de entender o operador ↔ é primeiro estabelecer uma proposição se, e então trabalhar com ela.

Considere essa proposição se:

> *Se* uma peruca está pendurada na cabeceira da cama do quarto de hóspedes, *então* tia Dóris está nos visitando.

Essa proposição declara que:

1. Se você vir uma peruca, então *sabe* que tia Dóris está aqui, *mas*
2. Se você vir tia Dóris, então *não pode ter certeza* de que há uma peruca.

Essa *proposição se* pode ser representada na LS como $W \rightarrow D$, com a seta apontando na direção da implicação: peruca *implica* em Dóris.

Agora, veja a seguinte proposição:

> Uma peruca estará pendurada na cabeceira da cama do quarto de hóspedes *se e somente se* tia Dóris estiver nos visitando.

Essa proposição é semelhante à anterior, mas amplia um pouco mais as coisas. Nesse caso, a proposição declara que:

1. Se você vir uma peruca, então *sabe* que tia Dóris está aqui, *e*
2. Se você vir tia Dóris, então *sabe* que há uma peruca.

Essa proposição pode ser representada na LS como $W \leftrightarrow D$, com a seta dupla, que dá a dica sobre seu significado: *tanto* peruca implica em Dóris *quanto* Dóris implica em peruca.

Já que o operador → também é chamado de condicional, o operador ↔ é chamado de *bicondicional.* Mas aqui o chamarei de *se e somente se* ou ↔.

Não confunda o operador se e somente se (↔) com o operador se (→).

Assim como ocorre com os outros operadores, para o operador ↔, você pode construir uma tabela com quatro linhas, abrangendo todas as possíveis combinações dos valores lógicos de *x* e *y:*

x	y	x↔y
V	V	V
V	F	F
F	V	F
F	F	V

Memorize a informação contida nessa tabela. Um forma rápida de decorar é: uma proposição do tipo ↔ (se...e...somente...se) apenas é verdadeira quando ambas as suas partes têm o mesmo valor verdade.

Uma característica importante da *proposição se e somente se* é que suas partes são *logicamente equivalentes*, o que significa que uma não pode ser verdadeira sem a outra.

Veja mais dois exemplos de *proposições se e somente se*:

> Você está no Rio de Janeiro se e somente se estiver na Lapa.
>
> Um número é par se e somente se puder ser dividido por dois sem que sobre resto.

A primeira proposição diz que Rio de Janeiro *é* a Lapa. A segunda proposição indica uma equivalência de suas duas partes: ser um número par é equivalente a ser divisível por dois.

Alguns livros sobre Lógica podem usar o símbolo ≡ como *operador se e somente se*, ao invés da seta dupla. Qualquer que seja o símbolo, o significado é o mesmo.

Como a LS É Parecida com a Aritmética Simples

Conforme visto no tópico “Os Cinco Operadores da LS”, anteriormente neste capítulo, a LS é prima da Matemática, já que os operadores de ambas as disciplinas assumem os valores que lhes são atribuídos para produzirem um novo valor. Mas as semelhanças entre elas não param por aí. Depois que você conhecer algumas outras semelhanças, a LS se tornará muito mais fácil de ser entendida.

Os valores de entrada e de saída

Na Aritimética, cada um dos operadores básicos transforma dois números em um. Por exemplo:

6 + 2 = **8** 6 – 2 = **4** 6 × 2 = **12** 6 ÷ 2 = **3**

Os dois números com que você começou são chamados de *valores de entrada* e o número que resulta dessa operação é chamado de *valor de saída.*

Em ambos os casos, a colocação do operador entre os dois valores de entrada (6 e 2) produz um valor de saída (em negrito no exemplo acima). Tendo em vista que há dois valores de entrada, essas operações são chamadas de *operações binárias.*

O sinal de menos (–) tem mais uma finalidade na Matemática. Quando você o coloca na frente de um número positivo ele o transforma em negativo. Mas quando o coloca na frente de um número negativo, ele o transforma em positivo. Por exemplo:

– –4 = 4

Nesse caso, o primeiro sinal de menos opera sobre um valor de entrada (– 4) e produz um valor de saída (4). Quando usado dessa forma, o sinal de menos é um *operador unário*, pois há somente um valor de entrada.

Na Aritmética, você tem que se preocupar com um número infinito de valores. Porém, na LS há somente dois valores: **V** e **F** (para saber mais sobre esses valores verdade, volte para o tópico “Valores lógicos” neste capítulo).

Como ocorre com a Aritmética, a Lógica tem quatro operadores binários e um operador unário. Na LS os operadores binários são $\wedge$, $\vee$, $\rightarrow$ e $\leftrightarrow$, e o operador unário é ~ (cada um desses operadores é tratado no tópico “Os Cinco Operadores da LS”, anteriormente neste capítulo).

LEMBRE-SE

As mesmas regras básicas se aplicam aos dois tipos de operadores, tanto na LS quanto na Aritmética:

- ✓ Coloque um operador binário entre qualquer par de valores de entrada e você obterá um valor de saída.
- ✓ Coloque um operador unário na frente de um valor de entrada e você obterá um valor de saída.

Por exemplo, começando com um par de valores de entrada, **F** e **V**, nessa ordem, você pode combiná-los usando os quatro operadores binários da seguinte forma:

$$\mathbf{F} \wedge \mathbf{V} = \mathbf{F} \qquad \mathbf{F} \vee \mathbf{V} = \mathbf{V} \qquad \mathbf{F} \rightarrow \mathbf{V} = \mathbf{V} \qquad \mathbf{F} \leftrightarrow \mathbf{V} = \mathbf{V}$$

Em todos os casos, o operador produz um valor de saída que será obviamente **V** ou **F**. Da mesma forma, se utilizar o operador unário ~ na frente de qualquer um dos valores de entrada **V** ou **F**, obterá um valor de saída:

$$\sim\mathbf{F} = \mathbf{V} \qquad \sim\mathbf{V} = \mathbf{F}$$

Não há substituto para a substituição

Mesmo com uma pequena noção de Álgebra, você sabe que letras podem servir de substitutas para os números. Se, por exemplo, eu lhe disser que

$a = 9$ e $b = 3$

você poderá concluir que

$a + b = 12 \qquad a - b = 6 \qquad a \times b = 27 \qquad a \div b = 3$

Quando você está trabalhando com constantes na LS, a mesma regra se aplica, precisando apenas substituir os valores corretos (**V** ou **F**) para cada constante; observe o exemplo a seguir:

Sabendo que *P* é verdadeira, *Q* é falsa e *R* é verdadeira, encontre os valores das seguintes proposições:

1. $P \vee Q$
2. $P \rightarrow R$
3. $Q \leftrightarrow R$

No problema 1 substitua **V** para *P* e **F** para *Q*. Isso resulta em **V** $\vee$ **F**, que é igual a **V**.

No problema 2 substitua **V** para *P* e **V** para *R.* Isso resulta em **V** $\rightarrow$ **V**, que é igual a **V**.

No problema 3 substitua **F** para *Q* e **V** para *R.* Isso resulta em **F** $\leftrightarrow$ **V**, que é igual a **F**.

Orientação para uso dos parênteses

Na Aritmética os parênteses são utilizados para agrupar números e operações. Por exemplo:

$-((4 + 8) \div 3)$

Nessa expressão os parênteses indicam que você deve resolver primeiro a equação 4 + 8, cujo resultado é 12. Então, no próximo conjunto de parênteses, resolva 12 ÷ 3, que resulta em 4. Finalmente, o operador unário de negação (–) altera o resultado para – 4.

Em geral você deve começar com o par de parênteses mais interno e passar aos seguintes, de dentro para fora. A LS usa os parênteses da mesma maneira. Por exemplo, considerando que *P* é verdadeira, *Q* é falsa e *R* é verdadeira, encontre os valores da seguinte proposição:

$$\sim((P \vee Q) \to \sim R)$$

Começando pelo conjunto de parênteses interno, $P \vee Q$ se torna **V** ∨ **F**, que, simplificada, fica **V**. Em seguida, passando para o próximo conjunto de parênteses, **V** → ~R se torna **V** → **F** que, simplificada, fica **F**. Finalmente, o ~ do lado de fora de todos os parênteses altera **F** para **V**.

O processo de redução da proposição com mais de um valor até um único valor é chamado de *Cálculo Proposicional*. Esse é um valioso instrumento sobre o qual você aprenderá mais no Capítulo 5.

Perdido na Tradução

Considerando que a LS é uma linguagem, depois que você conhecer suas regras, poderá traduzi-la para o português,... ou para o inglês, ou para espanhol e vice-versa, mas atenha-se ao português (porém, caso queira aprender espanhol ou inglês, leia *Espanhol Para Leigos* e *Inglês Para Leigos*).

A principal força da LS é que ela é, ao mesmo tempo, clara e não ambígua. Essas qualidades fazem com que a tradução da LS para o português se torne mais fácil. É por isso que chamo essa direção da tradução (português → LS) de *forma mais fácil*. A língua portuguesa, porém, pode ser confusa e ambígua (veja o tópico “Cavando em busca do ou”, no início deste capítulo, para uma demonstração de como uma palavra simples como *ou* pode ter inúmeros significados, dependendo de como é usada). Como você tem que tomar muito cuidado ao traduzir frases do português para LS, chamo essa direção da tradução (LS → português) de *forma não tão fácil*.

Saber traduzir as proposições em ambas as direções ampliará sua compreensão da LS, pois tornará muito mais claros os conceitos por trás de todos esses pequenos e estranhos símbolos. À medida que você avançar pelos próximos capítulos e começar a ficar confuso, lembre-se de que toda proposição na LS, não importa sua complexidade, pode ser reproduzida em português.

A forma mais fácil — traduzindo da LS para o português

Algumas vezes os exemplos são a melhor forma de se entender algo. Em vista disso, aqui vão alguns exemplos de várias maneiras de traduzir cada um dos tipos de operadores. São todos muito diretos, o que lhe permite escolher qual quer utilizar. Ao longo deste tópico usarei as seguintes constantes:

Considere *A* = Augusto ama Alma.

Considere *B* = O barco está na baía.

Considere *C* = Camila está pescando bagre.

Traduzindo proposições com ~

Você pode traduzir a proposição ~*A* para a língua portuguesa de qualquer das seguintes maneiras:

Não é o caso que Augusto ama Alma.

Não é verdade que Augusto ama Alma.

Augusto *não ama* Alma.

Augusto *não* sente amor por Alma.

Traduzindo proposições com ^

Veja dois modos de traduzir a proposição de LS *A* ^ *B*:

Augusto ama Alma *e* o barco está na baía.

Tanto Augusto ama Alma *quanto* o barco está na baía.

Traduzindo proposições com v

Observe as duas formas de traduzir a proposição *A* v *C:*

Augusto ama Alma *ou* Camila está pescando bagres.

Ou Augusto ama Alma *ou* Camila está pescando bagres.

Traduzindo proposições com →

Pode-se traduzir a proposição *B* → *C* em quaisquer das seguintes formas:

Se o barco está na baía, *então* Camila está pescando bagres.

O barco está na baía, *o que implica em* Camila estar pescando bagres.

O barco está na baía, *quer dizer que* Camila está pescando bagres.

O barco estará na baía *somente se* Camila estiver pescando bagres.

Traduzindo proposições com ↔

Na verdade, só existe um modo de traduzir a proposição de LS $C \leftrightarrow A$:

Camila estará pescando bagres *se e somente se* o barco estiver na baía.

Traduzindo proposições mais complexas

Para proposições mais complexas dê uma olhada no guia já mostrado neste capítulo, no tópico “Como a LS É Parecida com a Aritmética Simples”. Traduza as proposições, passo a passo, começando pela parte que estiver dentro dos parênteses. Por exemplo:

$(\sim A \wedge B) \vee \sim C$

A parte que está dentro dos parênteses é $(\sim A \wedge B)$, que pode ser traduzido para:

Augusto não ama Alma e o barco está na baía.

Adicionando a última parte da proposição, teremos:

Augusto não ama Alma e o barco está na baía *ou* Camila não está pescando bagres.

Observe que embora as frases estejam tecnicamente corretas, estão um tanto confusas, pois os parênteses foram removidos e todas elas estão juntas. Uma boa maneira de torná-las mais claras é traduzi-las da seguinte forma

Ou Augusto não ama Alma e o barco está na baía *ou* Camila não está pescando bagres.

A palavra *ou* antes da oração determina o quanto está abrangido pelo outro *ou* na parte final da mesma.

$\sim A \wedge (B \vee \sim C)$

pode ser traduzida como

Augusto não ama Alma e *ou* o barco está na baía *ou* Camila não está pescando bagres.

Observe, agora, um outro exemplo:

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

Começando pelos parênteses internos, tem-se $(\sim B \wedge C)$ como:

O barco não está na baía e Camila está pescando bagres.

Passando para os próximos parênteses, $(A \rightarrow (\sim B \wedge C))$ se torna:

Se Augusto ama Alma, então tanto o barco não está na baía *quanto* Camila está pescando bagres.

Observe que a adição das palavras *tanto e quanto* torna evidente que a proposição-*e* original está em parênteses. Por fim, adicione o ~ para obter:

Não é o caso que se Augusto ama Alma, então tanto o barco não está na baía quanto Camila está pescando bagres.

Concordo que é um pouco difícil, mas faz tanto sentido quanto uma frase assim poderia fazer. Provavelmente, você nunca terá que traduzir proposições muito mais complicadas que essa, mas, ainda assim, deve pensar um pouco no fato de que a LS é capaz de lidar com proposições de qualquer tamanho com perfeita clareza.

A forma "não tão fácil" – traduzindo do português para a LS.

Cada um dos quatro operadores binários da LS ($\wedge$, $\vee$, $\rightarrow$ e $\leftrightarrow$) é um conector que junta um par de proposições. Em português, palavras que conectam as proposições são chamadas de *conjunções*. Veja alguns exemplos de conjunções:

embora	se...então	ou
e	ou...ou	então
mas	nem...nem	portanto

entretanto todavia porém

se e somente se somente se

LEMBRE-SE

O operador ~ geralmente é traduzido para a língua portuguesa como *não*, mas também pode aparecer disfarçado sob outras formas, tais como nunca, nem, entre outros. Com todas essas variações na linguagem, uma lista completa de todas as possibilidades de tradução do português para a LS seria quatro vezes maior do que este livro. Veja, a seguir, apenas algumas dessas variações. Neste tópico, listo apenas algumas das palavras e frases mais comuns e examinarei cada uma delas, dando um exemplo de como traduzi-las para a LS.

Entretanto, primeiro é preciso definir algumas das constantes:

Considere *K* = Clara mora no Rio de Janeiro.

Considere *L* = Clara mora em Paraty.

Considere *M* = Eu gosto de Milena.

Considere *N* = Eu gosto de Núbia.

Considere *O* = Clara gosta de Olivia.

Mas, porém, entretanto, embora, todavia...

Muitas palavras da língua portuguesa ligam as proposições e têm o mesmo significado lógico da palavra *e*. Observe alguns exemplos:

Eu gosto de Milena, *mas* eu gosto de Núbia.

Embora eu goste de Milena, gosto de Núbia.

Gosto de Milena, todavia gosto de Núbia.

Eu gosto de Milena, *entretanto* gosto de Núbia.

Eu gosto de Milena, *porém* gosto de Núbia.

Cada uma dessas palavras carrega uma sutil diferença de significado, mas, na Lógica, você pode interpretá-las como tendo um só

$M \wedge N$

LEMBRE-SE

Depois que uma proposição foi traduzida da língua portuguesa para a LS, as regras da LS é que passam a ser aplicadas. Então, aqui como em qualquer proposição $\wedge$, se M ou N for falsa, a proposição $M \wedge N$ será falsa também. Caso contrário, será verdadeira.

Nem ...nem

A expressão *nem ...nem* nega ambas as partes da proposição. Por exemplo:

> Não gosto *nem* da Milena *nem* da Núbia.

Essa proposição significa as duas coisas: não gosto de Milena *e* não gosto de Núbia. Essa proposição é traduzida para a LS como

$\sim M \wedge \sim N$

Não...ambos

A expressão *não...ambos* significa que mesmo quando a proposição é negada como um todo, cada uma de suas partes pode não estar sendo negada. Por exemplo:

> *Não* gosto *de ambas*, Milena e Núbia.

Essa proposição declara que mesmo que eu não goste de ambas as mulheres juntas, posso gostar de uma delas. Então a proposição é traduzida para a LS como

$\sim(M \wedge N)$

…Se...

Você já sabe como traduzir uma proposição que começa com a palavra *se*. Mas isso se torna um pouco mais complicado quando a palavra *se* está inserida no meio da proposição, como a seguir:

> Eu gosto de Milena *se* Clara gosta de Olivia.

Para esclarecer a proposição basta desembaralhá-la, da seguinte forma:

Se Clara gosta de Olivia, eu gosto de Milena.

Com a proposição reorganizada, você pode ver que uma forma de traduzi-la é:

$O \rightarrow M$

...Somente se...

Essa expressão parece complicada até você perceber que, na verdade, é muito fácil e que não terá problema algum com ela. Primeiro observe a seguinte proposição verdadeira:

Clara mora em Paraty *somente se* vive no Rio de Janeiro.

Essa proposição faz sentido, pois a única forma de Clara morar em Paraty é se ela morar no Rio de Janeiro. Agora observe a seguinte proposição que também é verdadeira:

Se Clara mora em Paraty, então ela mora no Rio de Janeiro.

Essa proposição demonstra que as duas frases são logicamente equivalentes. Então, quando você encontrar um ...*somente se*... entre duas partes da proposição, perceba que ela é uma *proposição se* na ordem correta. Traduzida como

$L \rightarrow K$

...Ou...

Como já mencionei no tópico “Cavando em busca do ou”, no início deste capítulo, essa pequena palavra é um enorme problema. Veja o exemplo a seguir:

Clara mora no Rio de Janeiro *ou* Clara gosta de Olivia.

Dependendo de como a palavra *ou* é usada, ela pode ter dois significados:

Clara mora no Rio de Janeiro *ou* Clara gosta da Olivia, *ou ambos.*

Clara mora no Rio de Janeiro *ou* Clara gosta de Olivia, *mas não ambos.*

Tendo em vista as múltiplas facetas do *ou*, meu conselho é: quando você se deparar com um *ou*, aparentemente solitário em uma proposição a ser traduzida, isso provavelmente significa que alguém (como o seu professor) quer se certificar de que você saiba que, na Lógica, *ou* sempre significa *ou... ou ambos*. Então a tradução da proposição é

$K \vee O$

...Ou...ou ambos

Essa estrutura é fácil e clara: ela diz exatamente o que quer dizer. Por exemplo:

> Clara mora no Rio de Janeiro *ou* Clara gosta de Olivia, *ou ambos.*

Traduza essa proposição da seguinte forma:

$\mathrm{K} \vee \mathrm{O}$

...Ou... mas não ambos

Essa expressão tem o significado bem claro, mas não é tão fácil de ser traduzida. Por exemplo:

> Clara mora no Rio de Janeiro *ou* Clara gosta de Olívia, *mas não ambos.*

Para traduzir as palavras *mas não ambos* para a LS, você precisa desenvolver algumas etapas lógicas. Como já examinado no tópico "Mas, porém, entretanto, embora, todavia...", no início deste capítulo, a palavra *mas* torna-se $\wedge$ e as palavras *não ambos* são traduzidas aqui como $\sim(K \wedge O)$. Juntando tudo isso, a proposição inteira é traduzida como

$(K \vee O) \wedge \sim(K \wedge O)$

Capítulo 5
O Valor da Valoração

Neste Capítulo

- ► A valoração das proposições na LS
- ► Identificando o operador principal de uma proposição
- ► Conhecendo as oito formas proposicionais da LS

As pessoas amam a simplicidade.

Alguma vez você já leu a metade de uma crítica de um filme e pulou para o final para saber se era positiva ou negativa? Já folheou uma revista de automóveis para saber qual a avaliação que cada um dos modelos recebeu? Tenho certeza de que nunca se sentou com um amigo para classificar as pessoas que vocês conhecem em notas de 0 a 10.

Filmes, carros, homens e mulheres são complicados. Há tanto para se compreender. Porém, as pessoas amam a simplicidade. Tenho certeza de que você, assim como eu, sente certo alívio quando consegue reduzir a complexidade das coisas até algo bem pequeno que possa ser carregado no bolso.

A Lógica foi inventada com esse propósito em mente. No Capítulo 4, você descobriu como transformar uma proposição complicada em português para a linguagem da LS, usando apenas alguns símbolos. Neste capítulo dará um passo adiante, descobrindo como pegar uma proposição complicada da Lógica Formal e reduzi-la até um único valor lógico: **V** ou **F**. Assim como no caso da avaliação positiva ou negativa de uma crítica de cinema, a vida não pode ficar mais simples do que isso.

Esse processo de conversão é chamado de *valoração* ou *atribuição do valor lógico da proposição.* Qualquer que seja o termo usado, essa é uma das principais habilidades necessárias para o estudo da Lógica. Depois que você dominar esse processo, várias portas se abrirão magicamente.

O Valor É a Chave de Tudo

Um importante aspecto da LS é que ela permite que você simplifique proposições complexas através do processo de *valoração.* Quando você valora uma proposição na LS, substitui todas as suas constantes com valores lógicos (V e F), e depois reduz essa proposição a um *único valor lógico.* Quando pensar em *valoração*, tenha em mente o significado: *encontrar o valor de algo.*

LEMBRE-SE Valorar proposições corretamente é provavelmente a habilidade mais importante para se aprender com o estudo da Lógica. Alunos que têm problema com essa habilidade sofrem por dois motivos:

- ✓ Gastam muito tempo e ficam frustrados quando não sabem como realizar o processo de forma correta.
- ✓ Se constitui no primeiro passo que você precisa saber para lidar com uma série de outros processos, como verá mais adiante.

Porém, tenho boas notícias: valorar é uma habilidade de “inserir e encaixar” que não requer esperteza nem engenhosidade. Você precisa apenas conhecer as regras do jogo e depois praticar, praticar e praticar.

LEMBRE-SE As regras do jogo de valoração de proposições da LS são bem parecidas com as regras que você já conhece para valorar as proposições aritméticas (veja o Capítulo 4, onde descrevo as semelhanças entre a LS e a Aritmética). Por exemplo, observe este simples problema aritmético:

$$5 + (2 \times (4 - 1)) = ?$$

Para resolver o problema, primeiro avalie o que está dentro dos parênteses internos. Como o resultado de $4 - 1$ é 3, você pode substituir o valor de $(4 - 1)$ por 3 e o problema passa a ficar assim:

$$5 + (2 \times 3) = ?$$

Depois avalie o que está dentro do próximo conjunto de parênteses. Dessa vez tendo em vista que o valor de 2×3 é 6, uma nova substituição pode ser feita:

$$5 + 6 = ?$$

Nesse ponto o problema fica fácil de resolver. Já que $5 + 6$ resulta 11, essa será a resposta. Depois de uma série de *valorações*, uma quantidade de números e símbolos foi reduzida a um único *valor.*

Noções básicas da Valoração na LS

Observe o seguinte problema de LS:

Avalie a proposição $\sim(\sim P \rightarrow (\sim Q \wedge R)$

Aqui o objetivo é o mesmo de um problema aritmético: você quer valorar a proposição dada, o que significa que precisa encontrar seu valor.

No problema aritmético visto anteriormente, você já sabia os valores dos quatro números (5, 2, 4 e 1). Porém, no problema de LS precisa saber os valores de P, Q e R, isto é, precisa *interpretar* a proposição.

Interpretação de uma proposição é um conjunto fixo de valores lógicos para todas as constantes daquela proposição.

Por exemplo, uma possível interpretação para a proposição é: P = **V**, Q = **F** e R = **V**.

Lembre-se de que *P, Q* e *R* são constantes, e que **V** e **F** são valores lógicos. De forma que, quando escrevo *P* = **V**, isso não significa que essas duas coisas sejam iguais. É uma indicação de que o valor lógico de *P* é **V**.

Você pode não saber ao certo se essa interpretação está correta, mas, ainda assim, pode resolver o problema com base nessa interpretação, ou seja, assumindo que ela é correta. Nesse caso o problema completo seria:

> Sob a interpretação de que P = V, Q = F e R= V, valore a proposição $\sim(\sim P \rightarrow (\sim Q \wedge R))$.

Agora você pode resolver o problema. A primeira coisa a fazer é substituir cada constante por seu valor lógico:

$$\sim(\sim \mathbf{V} \rightarrow (\sim \mathbf{F} \wedge \mathbf{V}))$$

DICA

Depois de substituir as constantes na proposição de LS por seus valores lógicos, tecnicamente você não tem mais uma proposição, e os mais puristas podem torcer o nariz para esse exemplo. Mas enquanto estiver aprendendo a valorar, transformar proposições de LS em expressões desse tipo é muito útil.

O segundo e terceiro operadores ~ estão diretamente relacionados a valores lógicos, o que os torna fáceis de valorar, pois o valor de **~V** é **F**, e o de **~F** é **V** (vide o Capítulo 4 para refrescar sua memória sobre como trabalhar com os operadores lógicos que encontrar neste capítulo). Você pode reescrever a expressão da seguinte forma:

$$\sim(\mathbf{F} \rightarrow (\mathbf{V} \wedge \mathbf{V}))$$

DICA

Os parênteses na LS funcionam do mesmo modo que na aritmética. Eles dividem a expressão para que fique mais claro o que você precisa descobrir HOJ1 primeiro. Nesse caso o conjunto de parênteses interno contém **V** ∧ **V**. Tendo em vista que o valor de **V** ∧ **V** é **V**, a expressão simplificada é:

~(**F**→**V**)

Agora avalie o que está no outro conjunto de parênteses. Considerando que o valor de **F** → **V** é **V**, a expressão é simplificada para:

~(**V**)

Nesse ponto é fácil visualizar que ~**V** é valorada como **F**, que é a resposta do problema. O resultado aqui é semelhante ao do problema aritmético: você começa com uma proposição complexa e a valora encontrando seu valor que, na Lógica, será sempre **V** ou **F**.

Edificando um novo método

O método da valoração usado no tópico anterior funciona para todas as proposições de LS, não importa o quão complexas elas sejam. No próximo exemplo uso o mesmo método com uma pequena modificação estética: ao invés de reescrever a equação inteira a cada passo, apenas vou agregando os valores verdade enquanto prossigo. Observe este novo problema:

> Avalie *~(~P ∧ (~Q ↔ R))* usando a interpretação *P* = **F**, Q = **V** e *R* = **V**.

O primeiro passo é substituir as constantes por seus valores verdade. No exemplo usado no tópico anterior, reescrevi toda a equação. Desta vez apenas coloque os valores verdade de cada uma das constantes diretamente abaixo dela:

$$\begin{array}{cccccccc} \sim(& \sim P & \wedge & (& \sim Q & \leftrightarrow & R &)) \\ & F & & & V & & V & \end{array}$$

Esse exemplo tem dois operadores ~ que precedem as constantes imediatamente. É muito fácil de trabalhar com eles: simplesmente coloque o valor correto sob cada operador. Como você pode ver abaixo, os novos valores são maiores e os valores sublinhados ao lado deles mostram de onde vieram esses valores.

$$\begin{array}{cccccccccccc} \sim & (& \sim & P & \wedge & (& \sim & Q & \leftrightarrow & R &) &) \\ & & V & \underline{F} & & & F & \underline{V} & & \underline{V} & & \end{array}$$

Nesse ponto não tente valorar nenhum operador ~ que estiver imediatamente antes de um parênteses de abertura. Tendo em vista que eles negam *tudo o* que está dentro dos parênteses você terá que esperar até que saiba o valor de tudo o que está dentro dele antes de poder usar o operador.

Agora você pode trabalhar com o que está dentro dos parênteses. Comece pelo conjunto interno. O operador que está valorando aqui é o ↔. De um lado dele tem-se que o valor de ~Q é **F**. Do outro lado tem-se que o valor de *R* é **V**. Isso lhe dá **F** ↔ **V**, que é valorado como F. Coloque esse valor diretamente sob o operador que foi valorado, o ↔. Ao fazer isso, você poderá ver que o valor de tudo o que está dentro do parênteses é **F**:

$$\begin{array}{cccccccccccc} \sim & (& \sim & P & \wedge & (& \sim & Q & \leftrightarrow & R &) &) \\ & & V & F & & & \underline{F} & V & F & \underline{V} & & \end{array}$$

Agora passe para o próximo conjunto de parênteses. Aqui você vai valorar o ^. De um lado dele o valor de *~P* é **V**. Do outro lado tem-se que o valor de tudo o que está dentro dos parênteses internos (isto é, o valor do operador ↔) é **F**. Isso lhe dá **V** ^ **F**, que é valorado como **F**. Coloque esse valor sob o ^:

$$\begin{array}{cccccccccccc} \sim & (& \sim & P & \wedge & (& \sim & Q & \leftrightarrow & R &) &) \\ & & \underline{V} & F & F & & F & V & \underline{F} & V & & \end{array}$$

A última etapa é a valoração da proposição inteira. O operador que você vai valorar agora é o ~. Ele nega tudo o que está entre parênteses, isto é, o valor sob o ^, que é valorado como **F**. Coloque esse valor sob o ~:

~(~P ∧ (~Q ↔ R))
V V F <u>F</u> F V F V

Agora que tudo está valorado, esse último valor **V** é o de toda a proposição. Em outras palavras, quando *P* = **F**, *Q* = **V** e *R* = **V**, a proposição ~(~P ^ *(~Q* ↔ R)) é **V.** Como você pode ver, a valoração permite que transforme uma grande quantidade de informações em um único valor lógico. Não dá para ficar mais simples do que isso!

Construindo uma Proposição

Agora que você teve uma amostra da valoração, vai dar uma olhada no funcionamento das proposições na LS. Depois que você entender um pouco mais sobre as proposições, descobrirá que elas praticamente se valoram sozinhas.

Identificando subproposições

Uma *subproposição* é qualquer parte de uma proposição que consegue formar uma proposição completa por si só.

Por exemplo, a proposição $P \vee (Q \wedge R)$ contém as seguintes subproposições que conseguem formar uma nova proposição inteira:

> ✓ $Q \wedge R$
>
> ✓ P

Observe agora um outro pedaço da proposição $P \vee (Q \wedge R)$ que *não* é uma subproposição: $\vee (Q \wedge$. Muito embora esse também seja um pedaço da proposição, obviamente não é uma proposição inteira por si só. É apenas uma série de símbolos de LS sem qualquer sentido (no Capítulo 13 você descobrirá os detalhes sobre o que distingue uma proposição de um simples conjunto de símbolos).

Quando você valora uma proposição, começa pela valoração das menores subproposições possíveis, ou seja, das constantes individuais.

Para valorar a proposição *P* ∨ *(Q* ∧ *R)* baseado na interpretação *P* = **V**, *Q* = **V** e *R* = **F**, você começa pelo posicionamento do valor lógico de cada uma das constantes:

$$\begin{array}{ccccc} P & \vee & (Q & \wedge & R) \\ \mathbf{V} & & \mathbf{V} & & \mathbf{F} \end{array}$$

Agora pode valorar a subproposição maior *Q* ∧ *R:*

$$\begin{array}{ccccc} P & \vee & (Q & \wedge & R) \\ \mathrm{V} & & \underline{\mathrm{V}} & \mathbf{F} & \underline{\mathrm{F}} \end{array}$$

Por fim valore toda a proposição:

$$\begin{array}{ccccc} P & \vee & (Q & \wedge & R) \\ \underline{\mathrm{V}} & \mathbf{V} & \mathrm{V} & \underline{\mathrm{F}} & \mathrm{F} \end{array}$$

Como pode ver, a valoração de uma proposição longa funciona melhor quando você a desmembra pedaço a pedaço, em subproposições que podem ser valoradas mais facilmente.

Investigando uma proposição

Uma vez que você sabe o que é uma subproposição, fica fácil compreender o escopo do operador. O *escopo* de um operador é a menor subproposição que inclui o operador em questão.

Por exemplo, veja a proposição (*P* → (*Q* ∨ *R*)) ↔ *S. Suponha que você queira* descobrir o escopo do operador ∨. Duas das possíveis subproposições que incluem esse operador são *P* → (*Q* ∨ *R*) e *Q* ∨ *R*. A menor delas é *Q* ∨ *R*, então esse é o escopo do operador ∨.

Você também pode pensar no escopo de um operador como sendo a *área de influência* que ele tem na proposição.

Para ilustrar a área de influência, o escopo do operador ∨ aparece sublinhado na proposição a seguir:

$$(P \rightarrow (\underline{Q \vee R})) \leftrightarrow S$$

Esse exemplo mostra que o operador ∨ afeta ou influencia as constantes *Q* e *R*, mas *não* as constantes *P* ou *S*.

Na próxima ilustração sublinhei o escopo do operador → na mesma proposição:

$$(\underline{P \rightarrow (Q \vee R)}) \leftrightarrow S$$

Esse exemplo mostra que a área de influência do operador → inclui a constante *P* e a subproposição (*Q* ∨ *R*), mas não a constante *S.*

Antes de poder valorar um operador, você precisa saber o valor verdade de todas as outras constantes e do operador inserido em seu escopo. Uma vez que você saiba como encontrá-lo, fica fácil ver por que precisa começar a valoração pelo que está entre parênteses.

Por exemplo, na proposição (*P*→ (*Q* ∨ *R*)) ↔ *S,* o operador ∨ está dentro do escopo do →. Isso significa que você não pode valorar o → até que saiba o valor do ∨.

Seja cuidadoso ao tentar descobrir o escopo dos operadores nas proposições com ~. O escopo do ~ é sempre a menor subproposição que o segue imediatamente. Quando ele aparece na frente de uma constante, seu escopo inclui somente aquela constante. Imagine-o como estando ligado a essa constante. Observe, por exemplo, o escopo do primeiro operador ~ sublinhado na proposição abaixo:

$$\underline{\sim P} \wedge \sim(Q \wedge R)$$

Em contrapartida, quando um operador ~ está diante de um parêntese de abertura, seu escopo é tudo que está dentro do conjunto de parênteses. Observe o escopo do segundo operador ~, sublinhado na proposição abaixo:

$$\sim P \wedge \underline{\sim(Q \wedge R)}$$

Você poderia, da mesma forma, sublinhar o escopo do operador $\vee$ na proposição $\sim(P \vee Q)$ da seguinte forma:

$\underline{\sim(P \vee Q)}$ ERRADO!

Nesse caso o operador ~ está fora do parênteses e, portanto, fora do escopo do operador $\vee$.

$\sim\underline{(P \vee Q)}$ CERTO!

Quando valorar essa proposição, *primeiro* valore a subproposição $P \vee Q$ e depois a proposição inteira.

Atração principal: Encontrando os operadores principais

O *operador principal* é o mais importante de todos em uma proposição, pelos seguintes motivos:

- ✓ **Toda proposição na LS tem apenas um operador principal.**
- ✓ **O escopo do operador principal é a proposição inteira.** Assim, o operador principal afeta todas as outras constantes e operadores na proposição.
- ✓ **O operador principal é o último operador que você valora.** Isso faz sentido quando você pensa um pouco a respeito: se o escopo do operador principal é *todo o resto* da proposição, você precisa valorar essa parte antes que possa valorá-lo.

> Por exemplo, para valorar a proposição $(P \rightarrow (Q \leftrightarrow R)) \wedge S$ sob determinada interpretação, para obter o valor do operador $\leftrightarrow$, primeiro você precisa valorar a subproposição $Q \leftrightarrow R$. Isso lhe permite valorar $P \rightarrow (Q \leftrightarrow R)$ para obter o valor do operador $\rightarrow$. Por fim, pode valorar a proposição inteira, que lhe dará o valor do operador principal da proposição, o operador $\wedge$ (mostrarei como encontrar o operador principal mais adiante, neste tópico).
>
> ✓ **O valor do operador principal é o mesmo da própria proposição.**
>
> ✓ **O operador principal fica do lado de fora de todos os parênteses, *exceto* quando toda a proposição inclua um conjunto de parênteses extra (e removível).** Explicarei isso detalhadamente ainda neste tópico.

Tendo em vista a enorme importância do operador principal, você precisa ser capaz de reconhecê-lo em qualquer proposição. Isso geralmente é muito simples de ser feito, utilizando apenas algumas regras básicas. Toda proposição LS se enquadra em um desses três casos descritos a seguir. Se você encontrar alguma proposição que não se enquadre em um desses casos, é porque ela não está bem construída, o que significa que, na verdade, não é uma proposição. Esse tema será mais discutido no Capítulo 14. Por ora, tenha em mente que qualquer proposição que encontrar terá um operador principal que poderá ser facilmente encontrado.

Quando somente um operador está fora dos parênteses

Algumas vezes é fácil encontrar o operador principal, pois ele é o *único* operador fora dos parênteses. Por exemplo, observe a seguinte proposição:

$$(P \vee \sim Q) \wedge (R \rightarrow P)$$

Aqui o operador principal é o operador $\wedge$. Observe essa outra proposição:

$$\sim(P \wedge (Q \leftrightarrow R))$$

O operador principal aqui é o operador ~.

Quando não há operadores fora dos parênteses

Se *não* houver operadores fora dos parênteses, você terá que remover um conjunto interno. Por exemplo, na proposição a seguir, o conjunto de parênteses mais externo não é realmente necessário:

$$((\sim P \leftrightarrow Q) \rightarrow R)$$

Quando ele é retirado tem-se o seguinte:

$$(\sim P \leftrightarrow Q) \rightarrow R$$

Agora o único operador fora dos parênteses é o operador →, que é realmente o principal.

Neste livro evitei o uso de parênteses desnecessários, pois eles ocupam espaço e não acrescentam nada de útil à proposição. No Capítulo 14, discutirei os detalhes mais importantes dos motivos que levam uma proposição a conter parênteses extras.

Quando há mais de um operador fora dos parênteses

Algumas proposições têm mais que um operador fora dos parênteses. Por exemplo:

$$\sim(\sim P \rightarrow Q) \vee \sim(P \rightarrow Q)$$

Quando houver mais de um operador fora dos parênteses, o principal será sempre outro que *não* o ~.

LEMBRE-SE

No exemplo anterior, o operador principal é o $\vee$.

Oito Formas de Proposições da LS

Na LS uma variável pode substituir uma proposição inteira (ou subproposição). Você pode usar variáveis para classificar as proposições em oito diferentes *formas proposicionais,* que são versões gerais de proposições de LS. A Tabela 5-1 apresenta as oito formas básicas das proposições.

Tabela 5-1: As Oito Formas Proposicionais de LS

Positivas	***Negativas***
$x \wedge y$	$\sim(x \wedge y)$
$x \vee y$	$\sim(x \vee y)$
$x \rightarrow y$	$\sim(x \rightarrow y)$
$x \leftrightarrow y$	$\sim(x \leftrightarrow y)$

Para saber como essas proposições funcionam, observe os três exemplos a seguir, onde os operadores principais são todos $\wedge$.

$$P \underline{\wedge} Q$$

$$(P \vee \sim Q) \underline{\wedge} \sim(R \rightarrow S)$$

$$(((\sim P \leftrightarrow Q) \rightarrow R) \vee (\sim Q \wedge S))\underline{\wedge} R$$

Partes de um todo

Pessoalmente, acho que toda a terminologia utilizada para as várias partes de uma proposição é um pouco exagerada. Caso seu professor queira que você saiba todos esses termos, terá que decorá-los. Para mim, o mais importante é que quando você se deparar com uma proposição, saiba encontrar o operador principal e distinguir em qual das oito formas ela se enquadra. Quando houver necessidade de identificar as partes de uma proposição, basta dizer “primeira parte” e “segunda parte”.

Veja a seguir algumas regrinhas básicas:

✓ Quando uma proposição está na forma $x \wedge y$, é chamada de $\wedge$, ou de *conjunção*. Nesse caso, ambas as partes que a compõem são chamadas de *conjuntos.*

✓ Quando uma proposição está na forma $x \vee y$, é chamada de $\vee$ ou *disjunção*. Nesse caso, ambas as partes da proposição são chamadas de *disjuntos*.

✓ Quando uma proposição está na forma $x \rightarrow y$, é chamada de $\rightarrow$ ou de *implicação.* Nesse caso, a primeira parte é chamada de *antecedente* e a segunda de *consequente.*

✓ Quando uma proposição está na forma $x \leftrightarrow y$, é chamada de $\leftrightarrow$ ou *bi-implicação*.

Embora todas essas proposições sejam obviamente diferentes, você pode representar cada uma delas usando a seguinte forma de proposição:

$$x \wedge y$$

Por exemplo, na proposição $P \wedge Q$, a variável x substitui a subproposição P e a variável y substitui a subproposição Q. Do mesmo modo, na proposição $(P \vee {\sim}Q)\ \underline{\wedge}\ {\sim}(R \rightarrow S)$, o x substitui a subproposição $(P \vee {\sim}Q)$ e o y substitui a subproposição ${\sim}(R \rightarrow S)$. Por fim, na proposição $((({\sim}P \leftrightarrow Q) \rightarrow R) \vee ({\sim}Q \wedge S)) \wedge R$, o x substitui $((({\sim}P \leftrightarrow Q) \rightarrow R) \vee ({\sim}Q \wedge S))$ e o y substitui o R.

Quando o operador principal de uma proposição for um dos quatro operadores binários ($\wedge$, $\vee$, $\rightarrow$ ou $\leftrightarrow$), sua forma proposicional será uma das quatro positivas da Tabela 5-1. Entretanto, quando o operador principal da proposição for o operador ~, sua forma será uma das negativas na Tabela 5-1. Para descobrir qual das formas usar, você deve procurar pelo operador com o segundo maior escopo. Por exemplo:

$$\sim((P \rightarrow Q) \leftrightarrow (Q \vee R))$$

Nesse caso o operador principal é o ~. O operador com o segundo maior escopo é o operador ^, abrangendo tudo aquilo que está dentro dos parênteses. Então você pode representar essa proposição usando a seguinte forma proposicional:

$$\sim(x \leftrightarrow \mathrm{y})$$

Aqui a variável *x* está substituindo a subproposição ($P \rightarrow Q$) e a variável *y* substitui ($Q \vee R$).

Aprender a reconhecer a forma básica de determinada proposição é uma habilidade que você usará nos capítulos mais adiante. Por ora, basta saber que cada proposição pode ser representada por apenas uma das oito formas proposicionais básicas.

Revisitando a Valoração

Depois de conhecer os novos conceitos trazidos neste capítulo, você deve estar achando que a valoração faz muito mais sentido. Assim, é menos provável que cometa erros, pois você entendeu como todas as peças da proposição se encaixam.

Por exemplo, suponha que você queira valorar a proposição ~(~(*P* ∨ *Q*) ∧ (~*R* ↔ *S*)) sob a interpretação de que *P* = **V**, *Q* = **F** e *S* = **V**. Pode parecer complicado mas você deve encarar o desafio!

Antes de começar, observe a proposição. Ela está na forma ~(*x* ∧ *y*), com a primeira parte da proposição sendo ~(*P* ∨ *Q*) e a segunda sendo (~*R* ↔ *S*). Você precisa obter o valor lógico de ambas as partes da proposição antes que possa valorar o operador ∧. Somente depois disso é que poderá valorar o primeiro operador ~, que é o operador principal da proposição.

Comece inserindo os valores verdade sob as referidas constantes:

```
~(~(P ∨ Q) ∧ (~R ↔ S))
    V   F       F    V
```

Agora pode escrever o valor do operador ~, que aparece na frente da constante *R*:

```
~(~(P ∨ Q) ∧ (~R ↔ S))
    V   F      VF    V
```

Nesse ponto pode obter o valor dos operadores ∨ e ↔:

```
~(~(P ∨ Q) ∧ (~R ↔ S))
    V V F      VF V  V
```

Ainda que você fique tentado a valorar o operador ∧, primeiro precisa do valor da subproposição ~(*x* ∨ *y*), o que significa que precisa obter o valor do operador ~:

```
~(~(P ∨ Q) ∧ (~R ↔ S))
   FV V F      VF V  V
```

Agora sim, pode valorar o operador ^:

$$\begin{array}{cccccccccc} \sim(& \sim(P & \vee & Q) & \wedge & (\sim & R & \leftrightarrow & S)) \\ & \underline{F}\ V & V & F & F & V & F & \underline{V} & V \end{array}$$

Finalmente, depois de avaliar cada um dos outros operadores, você pode valorar o principal:

$$\begin{array}{cccccccccc} \sim(& \sim(P & \vee & Q) & \wedge & (\sim & R & \leftrightarrow & S)) \\ V & \underline{F}\ V & \underline{V} & F & F & V & F & \underline{V} & V \end{array}$$

O valor lógico do operador principal é o valor da proposição inteira, de forma que você pode concluir que, sob a interpretação dada, a proposição é verdadeira.

Capítulo 6

Caindo pelas Tabelas: Valorando Proposições com as Tabelas Verdade

Neste Capítulo

- Criando e analisando tabelas verdade
- Reconhecendo quando a proposição é tautologia, contradição ou contingência
- Entendendo equivalência semântica, consistência e validade

Neste capítulo, você descobrirá uma das mais importantes ferramentas na Lógica Sentencial (LS): a *tabela verdade*. As tabelas verdade lhe permitem valorar uma proposição sob qualquer interpretação possível, e isso, por sua vez, lhe permite tirar conclusões gerais sobre uma proposição, mesmo quando você não saiba os valores verdade dessas constantes.

Tabelas verdade abrem um vasto território novo na Lógica. Elas são, antes de mais nada, uma forma fácil para se descobrir se um argumento é válido ou inválido — uma questão central na Lógica. Porém, além disso, tabelas verdade tornam possível a identificação de *tautologias* e *contradições*: proposições em LS que são sempre verdadeiras ou falsas respectivamente.

Você também pode utilizar as tabelas verdade para decidir sobre a *consistência* de um conjunto de proposições, isto é, sobre a possibilidade de todas elas serem verdadeiras. Enfim, com as tabelas verdade você pode descobrir se duas proposições são *semanticamente equivalentes*, ou seja, se têm o mesmo valor verdade em todos os casos possíveis.

É aqui que a porca torce o rabo; aperte os cintos!

Colocando Tudo na Tabela: A Alegria da Força Bruta

Algumas vezes você precisa de astúcia e perspicácia para solucionar um problema. Antes de obter a resposta, precisa ter aquele momento “Aha!”, que lhe permitirá ver as coisas de um modo completamente novo. Momentos “Aha!” podem ser divertidos, mas também frustrantes, especialmente se você estiver estudando para fazer um exame, sob pressão, e o tal momento “Aha!” não aparece.

Tabelas verdade são o antídoto para esse “Aha!”. Elas se apoiam em um método que os matemáticos chamam informalmente de *força bruta*. Nesse tipo de abordagem, ao invés de tentar encontrar um dos caminhos dourados para o sucesso, você exaure, obstinadamente, todos os caminhos possíveis. Métodos de força bruta podem tomar muito tempo, mas, no fim das contas, você sempre encontrará a resposta que estava procurando.

Veja como isso funciona. Suponha que em seu primeiro exame de Lógica você se depare com a seguinte questão enigmática:

> O que você pode dizer sobre essa proposição: $P \rightarrow (\sim Q \rightarrow (P \wedge \sim Q))$? Justifique sua resposta.

Com o relógio correndo, você provavelmente pensaria em uma porção de coisas que gostaria de falar sobre essa proposição, mas isso não o ajudará nem um pouco na nota. Aí, olha bem para ela e espera o momento “Aha!” chegar. De repente ele surge, e você pensa na proposição da seguinte forma:

> Essa proposição está declarando o óbvio: “assumindo que *P* é verdadeira e que *Q* é falsa, posso concluir que *P* é verdadeira e que *Q* é falsa”. Em consequência, a proposição é *sempre* verdadeira.

O resultado não foi ruim. Mas, e se o momento “Aha!” nunca chegasse? Se você não soubesse como “justificar sua resposta”?

Pior de tudo, e se a questão fosse assim:

> O que você pode dizer sobre essa proposição: $((\sim P \vee Q) \rightarrow ((R \wedge \sim S) \vee T)) \rightarrow (\sim U \vee ((\sim R \vee S) \rightarrow T)) \rightarrow ((P \wedge \sim U) \vee (S \rightarrow T))$? Justifique sua resposta.

Esse é o momento de usar a força bruta, e as tabelas verdade são o seu passaporte.

Minha Primeira Tabela Verdade

A tabela verdade é uma forma de organizar informações na LS. Ela permite que você determine todas as combinações possíveis de valores para cada uma das constantes e veja o que acontece em cada um dos casos. Você já viu algumas pequenas tabelas verdade no Capítulo 4. Quando apresentei cada um dos cinco operadores da LS mostrei, através de uma tabela verdade, como interpretar todas as possibilidades de valores de entrada para obter o valor de saída.

No tópico seguinte mostrarei como montar, preencher e extrair as conclusões de uma tabela verdade trabalhando com a proposição $P \rightarrow (\sim Q \rightarrow (P \wedge \sim Q))$.

Montando uma tabela verdade

A tabela verdade é um modo de organizar, em linhas horizontais, *todas as interpretações possíveis* de uma proposição, permitindo que você a valore sob todas essas interpretações.

Montar uma tabela verdade é um processo simples, de quatro passos:

1. **Construa uma linha superior com cada uma das constantes do lado esquerdo e com a proposição do lado direito.**

P	Q	P	→	(~	Q	→	(P	∧	~	Q))

2. **Determine o número de linhas adicionais de que sua tabela necessita, baseado no número de constantes da proposição.**

   Uma tabela verdade precisa de uma linha para cada interpretação possível de determinada proposição.

Para descobrir de quantas linhas vai precisar basta multiplicar o número dois por si mesmo para cada constante da proposição. Considerando que a proposição $P \rightarrow (\sim Q \rightarrow (P \wedge \sim Q))$ tem duas constantes você precisará de quatro linhas.

Para se certificar de que está no caminho certo, verifique a Tabela 6-1, que contém o número de linhas necessárias para proposições com um número de constantes entre um e cinco.

Tabela 6-1: Números de Constantes e Linhas na Tabela Verdade

Números de Constantes	***Constantes***	***Número de Interpretações (Linhas na Tabela Verdade)***
1	P	2
2	P e Q	2 × 2 = 4
3	P, Q e R	2 × 2 × 2= 8
4	P, Q, R e S	2 × 2 × 2 × 2 = 16
5	P, Q, R, S e T	2 × 2 × 2 × 2 × 2 = 32

3. **Monte as colunas das constantes para que cada combinação possível de valores verdade conste da tabela.**

Uma boa maneira de preencher essas colunas com **V**s e **F**s é começar pela constante mais à direita, na coluna de constantes (nesse caso, a coluna *Q*), e preenchê-la alternando entre **V**s e **F**s **– V F VF** – até o final da tabela. Depois, passe uma coluna à esquerda e preencha-a, alternativamente, com **V**s e **F**s, de dois em dois – **VVFF**. Se houver mais colunas (por exemplo, se a proposição tiver três ou quatro constantes), continue alternando os **V**s e os **F**s de quatro em quatro (**VVVVFFFF**), de oito em oito (**VVVVVVVVFFFFFFFF**) e assim por diante.

Neste exemplo alterne **V F V F** sob a coluna do *Q* e, em seguida, **V V F F** sob a coluna do *P*:

P	Q	P	→	(~	Q	→	(P	∧	~	Q))
V	V									
V	F									
F	V									
F	F									

4. **Desenhe linhas horizontais separando as linhas e verticais separando as constantes de cada um dos operadores na proposição.**

P	Q	P	→	(~	Q	→	(P	∧	~	Q))
V	V									
V	F									
F	V									
F	F									

Esse passo é importante por três razões. Primeiro, a tabela começa a ficar mais definida e evita confusões. Segundo, você saberá que a tabela estará terminada quando todos os quadradinhos estiverem preenchidos com **V**s e **F**s. Terceiro, porque a tabela totalmente preenchida ficará clara e legível (se você desenhar as linhas corretamente, usando uma régua, amolecerá o coração até do professor mais frio).

Não é necessário desenhar colunas para os

parênteses, mas certifiquese de que cada um dos parênteses apareça com a constante ou com o operador que aparece imediatamente *depois* dele, exceto, é claro, aqueles no final da proposição.

Preenchendo uma tabela verdade

Cada linha da sua tabela verdade conta para uma das diferentes interpretações da proposição. O preenchimento da tabela verdade agora passa a ser um simples processo de valoração da proposição sob cada umas das interpretações possíveis (nesse caso, quatro).

No Capítulo 4, foi visto como valorar uma proposição de dentro para fora. As regras dessa etapa são as mesmas, mas agora você precisa trabalhar cada passo do processo, em cada uma das linhas da tabela.

O procedimento passo a passo que segue mostra como trabalhar com as colunas, porque é mais fácil e rápido do que valorar linha por linha.

Enquanto avanço, passo a passo, observe que cada uma das colunas que acabei de preencher aparece sublinhada e cada coluna usada durante cada etapa do processo aparece em negrito.

1. **Copie os valores de cada constante nas colunas da proposição adequadas para cada uma.**

P Q	P	→	(~	Q	→	(P	∧	~	Q))
V V	V			V		V			V
V F	V			F		V			F
F V	F			V		F			V
F F	F			F		F			F

Apenas copie. Muito fácil, não é?

2. **Em cada uma das colunas que tenha um operador ~ diretamente na frente de uma constante, escreva a negação correspondente daquela constante em cada uma das linhas.**

P Q	P	→	(~	Q	→	(P	∧	~	Q))
V V	V		F	V		V		F	V
V F	V		V	F		V		V	F
F V	F		F	V		F		F	V
F F	F		V	F		F		V	F

Certifique-se de que cada operador ~ esteja na frente de uma constante. Se estiver diante de um parênteses de abertura, ele nega todo o valor contido nos parênteses. Nesse caso, você terá que esperar até descobrir esse valor.

Como pode ver, essa etapa não é muito mais difícil do que a anterior.

3. **Começando pelo conjunto de parênteses mais interno, preencha a coluna diretamente abaixo do operador daquela parte da proposição.**

A Etapa 3 é realmente a essência da tabela verdade. A boa notícia é que com alguma prática você conseguirá preencher essa etapa da tabela com muita rapidez.

Os parênteses internos, nesse exemplo, contêm a proposição *P* ^ ~*Q*. O operador que está sendo usado nessa valoração é o ^, e os dois valores de entrada estão nas colunas sob a constante *P* e o operador ~.

Por exemplo, na primeira linha o valor sob *P* é **V** e o valor sob o operador ~ é **F**. Como **V** e **F** resultam em **F**, escreva esse valor na linha abaixo do operador ^.

Repita essa etapa para as outras três linhas e sua tabela ficará assim:

P	Q	P	→	(~	Q	→	(P	∧	~	Q))
V	V	V		F	V		V	**F**	F	V
V	F	V		V	F		V	**V**	V	F
F	V	F		F	V		F	**F**	F	V
F	F	F		V	F		F	**F**	V	F

4. **Repetindo a Etapa 3, trabalhe de dentro para fora do primeiro conjunto de parênteses até que tenha valorado o operador principal da proposição.**

   Passando para o próximo conjunto de parênteses, o operador que está sendo valorado é o → dentro dos parênteses externos. Os dois valores de entrada estão nas colunas sob o primeiro operador ~ e o operador ^.

   Por exemplo, na primeira linha, o valor sob o operador ~ é **F** e o valor sob o operador ^ é **F**. Como **F** → **F** resulta em **V**, escreva esse valor na linha imediatamente abaixo do operador →.

   Ao completar a coluna sua tabela ficará assim:

P	Q	P	→	(~	Q	→	(P	∧	~	Q))
V	V	V		F	V	**V**	V	F	F	V
V	F	V		V	F	**V**	V	V	V	F
F	V	F		F	V	**V**	F	F	F	V
F	F	F		V	F	**F**	F	F	V	F

Agora você está pronto para o operador principal, o →, fora dos parênteses (veja o Capítulo 5 para saber mais sobre como determinar o operador principal). Os dois valores de entrada estão nas colunas sob a constante *P* e sob o operador →.

Por exemplo, na primeira linha o valor sob *P* é **V** e o valor sob o operador → é **V**. Como **V** → **V** resulta em **V**, escreva isso diretamente sob o operador principal → .

Ao terminar a coluna ficará assim:

P	Q	P	→	(~	Q	→	(P	∧	~	Q))
V	V	V	**V**	F	V	V	V	F	F	V
V	F	V	**V**	V	F	V	V	V	V	F
F	V	F	**V**	F	V	V	F	F	F	V
F	F	F	**V**	V	F	F	F	F	V	F

A coluna sob o operador principal deve ser a última a ser preenchida. Se não foi assim que você fez, é melhor apagar tudo (você estava usando lápis, não é mesmo?) e refazer todas as etapas.

Interpretando uma tabela verdade

Circule a coluna inteira sob o operador principal para que essa informação se sobressaia no momento de interpretar a tabela. A coluna sob o operador principal é a mais importante da tabela, pois ela mostra o valor verdade de cada interpretação da proposição.

Por exemplo, se quiser saber qual a valoração da proposição quando P é falsa e Q é verdadeira, basta olhar a terceira linha da tabela. O valor contido nessa linha, na coluna do operador principal, é **V**, de forma que quando P é falsa e Q é verdadeira, a proposição é verdadeira.

Neste ponto e com enorme convicção, você pode voltar para a questão original:

> O que você pode dizer sobre a proposição: $P \rightarrow (\sim Q \rightarrow (P \wedge \sim Q))$? Justifique a resposta.

Com sua tabela verdade super confiável, você poderá dizer a seu professor exatamente o que ele quer ouvir sobre a proposição: que ela é sempre verdadeira, independentemente do valor das constantes P e Q.

Como justificar essa resposta? Você não terá que justificar, pois a tabela verdade faz isso por você. Desde que tenha sido preenchida corretamente, a tabela deixa um rastro de cada uma das possíveis interpretações da proposição. Nenhuma outra interpretação é possível, então o seu trabalho está terminado.

Colocando as Tabelas Verdade para Funcionar

Depois que souber como usar as tabelas verdade, você começará a entender a LS sob nova perspectiva. Neste tópico, mostro como lidar com alguns dos problemas mais comuns de proposições individuais, pares ou em conjuntos e dos argumentos (também mostro, nos capítulos mais adiante, como lidar com essas mesmas questões usando uma variedade de ferramentas diferentes).

Enfrentando as tautologias e as contradições

Toda proposição de LS se enquadra em umas destas três categorias: tautologias (verdadeiras sob qualquer interpretação), contradições (falsas sob qualquer interpretação) ou contingências (podem ser verdadeiras ou falsas, dependendo da interpretação).

No tópico “Minha Primeira Tabela Verdade”, no início deste capítulo, você viu como usar as tabelas verdade para determinar o valor lógico de uma proposição sob cada uma das interpretações possíveis, o que lhe permite dividir as proposições em três importantes categorias:

- ✓ **Tautologias**: *Tautologia* é uma proposição que é sempre verdadeira, não importa quais sejam os valores lógico das constantes. Um exemplo de tautologia é $P \vee \sim P$. Pois, se *P* oû*P* for verdadeira, ao menos uma das partes dessa proposição ou será verdadeira, então ela será sempre verdadeira.
- ✓ **Contradições:** *Contradição* é uma proposição que é sempre falsa, independentemente dos valores lógicos de suas constantes. Um exemplo de contradição é $P \wedge \sim P$. Pois, se *P* ou ~*P* for falsa, ao menos uma parte da proposição e será falsa, portanto, ela inteira será sempre falsa.

- ✓ **Contingências:** *Contingência* é uma proposição que é verdadeira sob pelo menos uma interpretação e falsa sob pelo menos outra interpretação. Um exemplo de uma contingência é $P \rightarrow Q$. Essa proposição será verdadeira quando P for verdadeira e Q for verdadeira, mas será falsa quando P for verdadeira e Q for falsa.

Não cometa o erro de pensar que toda proposição é uma tautologia ou uma contradição. Muitas contingências não se encaixam em nenhuma dessas categorias.

As tabelas verdade são a ferramenta ideal para decidir em que categoria determinada proposição se encaixa. Por exemplo, no tópico “Minha Primeira Tabela Verdade”, foi usada a tabela verdade para mostrar que a proposição $P \rightarrow (\sim Q \rightarrow (P \wedge \sim Q))$ é valorada como verdadeira em todas as linhas da tabela verdade, assim, ela é uma tautologia.

Da mesma forma, se uma proposição é valorada como falsa em todas as linhas da tabela verdade, ela é uma contradição. Por fim, se uma proposição é valorada como verdadeira em pelo menos uma das linhas e falsa em pelo menos uma das linhas, ela é uma contingência.

Julgando equivalência semântica

Quando analisa proposições individuais, você pode usar a tabela verdade para valorá-las sob todas as combinações possíveis de valores para cada uma de suas constantes. Agora, dê um passo à frente nesse processo e compare duas proposições ao mesmo tempo.

Quando duas proposições são *semanticamente equivalentes*, elas têm o mesmo valor verdade sob todas as interpretações.

LEMBRE-SE

Você já conhece um exemplo simples de duas proposições que são semanticamente equivalentes: *P* e ~~*P*. Quando o valor de *P* é **V**, o valor de ~~*P* também é **V**. Do mesmo modo, se o valor de *P* é **F**, o de ~~*P* também é **F**.

Esse exemplo é fácil de ser verificado, pois, com apenas uma constante, só há duas interpretações possíveis. Como você pode imaginar, quanto mais constantes tiver, mais complicada se tornará a equivalência semântica.

No entanto, uma tabela verdade pode ajudar. Por exemplo, as proposições $P \rightarrow Q$ e $\sim P \vee Q$ são semanticamente equivalentes? Você pode descobrir a resposta construindo a tabela verdade para as duas proposições.

P	Q	P	→	Q	~	P	∨	Q
V	V							
V	F							
F	V							
F	F							

Conforme já visto no tópico “Minha Primeira Tabela Verdade”, o primeiro passo para preencher a tabela é copiar os valores de cada constante nas devidas colunas:

P	Q	P	→	Q	~	P	∨	Q
V	V	V		V		V		V
V	F	V		F		V		F
F	V	F		V		F		V
F	F	F		F		F		F

Depois disso, cuide de quaisquer operadores ~ que se apliquem diretamente sobre uma constante:

P	Q	P	→	Q	~	P	∨	Q
V	V	V		V	F	V		V
V	F	V		F	F	V		F
F	V	F		V	V	F		V
F	F	F		F	V	F		F

Em seguida, complete as valorações de ambas as proposições, separadamente, da mesma forma que você faria com apenas uma proposição:

P	Q	P	→	Q	~	P	∨	Q
V	V	V	V	V	F	V	V	V
V	F	V	F	F	F	V	F	F
F	V	F	V	V	V	F	V	V
F	F	F	V	F	V	F	V	F

Quando duas proposições têm o mesmo valor verdade em todas as linhas da tabela verdade, elas são semanticamente equivalentes. Caso contrário, não.

A tabela mostra que, nesse caso, as duas proposições são semanticamente equivalentes. O Capítulo 8 mostra como esse importante conceito de equivalência semântica é aplicado em provas de LS.

Mantendo a consistência

Se você pode comparar duas proposições ao mesmo tempo, por que não mais do que duas?

Quando um conjunto de proposições é *consistente*, ao menos uma interpretação as torna todas verdadeiras. Quando um conjunto de proposição é *inconsistente*, nenhuma interpretação faz com que todas elas sejam verdadeiras.

Por exemplo, observe o seguinte conjunto de proposições:

$P \vee \sim Q$

$P \rightarrow Q$

$P \leftrightarrow \sim Q$

É possível que todas as três proposições sejam verdadeiras ao mesmo tempo? Isto é, existe qualquer combinação de valores lógicos para as constantes *P* e *Q* que faça com que todas as três proposições sejam valoradas como verdadeiras?

Novamente, a tabela verdade é a ferramenta de escolha. Desta vez, no entanto, coloque todas as três proposições na tabela verdade. Tomei a liberdade de preencher as informações corretas para a primeira proposição. Primeiro, copiei os valores de *P* e *Q* para cada uma das colunas correspondentes. Depois, determinei o valor de ~*Q*. Por fim, valorei a proposição *P* ∨ ~*Q* inteira, inserindo o valor de cada linha sob o operador principal.

Em três dessas quatro linhas, a proposição é valorada como verdadeira. Mas, quando *P* é verdadeira e *Q* é falsa, a proposição é falsa. Já que você está buscando por linhas onde todas as três proposições sejam verdadeiras, pode descartar a linha onde a primeira proposição é falsa.

P	Q	P	∨	~	Q	P	→	Q	P	↔	~	Q
V	V	V	V	F	V							
V	F	V	V	V	F							
F	V	F	F	F	V	—	—	—	—	—	—	—
F	F	F	V	V	F							

DICA

Quando preencher uma tabela verdade para testar sua consistência, você deve seguir verticalmente, como de costume, mas valore uma proposição de cada vez. Quando encontrar uma linha onde a proposição resulte como falsa, desenhe traços até o final. Isso serve para lembrar de não valorar mais nenhuma outra proposição nessa linha.

Repetindo esse processo para as duas outras proposições dadas, a tabela ficará assim:

P	Q	P	∨	~	Q	P	→	Q	P	↔	~	Q
V	V	V	V	F	V	V	V	V	V	F	F	V
V	F	V	V	V	F	V	F	F	—	—	—	—
F	V	F	F	F	V	—	—	—	—	—	—	—
F	F	F	V	V	F	F	V	F	F	F	V	F

[1]

1 N.E.: Assim, chegamos à conclusão que nenhuma linha da tabela verdade com os resultados das 3 proposições possui os três valores lógicos iguais a V. Logo o conjunto de proposições é inconsistente.

LEMBRE-SE

Quando cada linha da tabela verdade tiver ao menos uma proposição que resulte como falsa, as proposições são inconsistentes. Caso contrário, são consistentes.

Neste caso, você sabe que as três proposições são inconsistentes, pois, de acordo com todas as interpretações, pelo menos uma proposição é falsa.

Argumentando com a validade

Como visto no Capítulo 3, em um argumento válido, quando todas as premissas são verdadeiras, a conclusão também o é. Aqui, essa mesma ideia básica está descrita em termos de interpretação:

LEMBRE-SE

Quando um argumento é *válido*, não existe interpretação onde todas as premissas sejam verdadeiras e a conclusão seja falsa. Quando um argumento é *inválido*, porém, há ao menos uma interpretação onde todas as premissas sejam verdadeiras e a conclusão seja falsa.

Você também pode usar a tabela verdade para decidir se um argumento inteiro é válido. Por exemplo, observe o seguinte argumento:

Premissas:

$P \wedge Q$

$R \rightarrow \sim P$

Conclusão:

$\sim Q \leftrightarrow R$

Nesse caso, o argumento tem três constantes — *P, Q* e *R* —, então, a tabela verdade precisa de oito linhas, pois 2 × 2 × 2 = 8 (veja a Tabela 6-1).

LEMBRE-SE

Para montar uma tabela verdade grande: comece com a coluna da constante mais à direita (*R* nesse caso) e escreva **V**, **F**, **V**, **F** e assim por diante, alternando todas as outras linhas até o fim. Então, passe para a coluna do lado esquerdo e escreva **V**, **V**, **F**, **F** e assim por diante, alternando as linhas de dois em dois. Continue se movendo para o lado esquerdo e dobrando as proporções, alternando a próxima de quatro em quatro, depois de oito em oito e assim por diante, até que a tabela esteja toda preenchida.

A tabela verdade ficará assim:

P	Q	R	P	∧	Q	R	→	~	P	~	Q	↔	R
V	V	V											
V	V	F											
V	F	V											
V	F	F											
F	V	V											
F	V	F											
F	F	V											
F	F	F											

No tópico "Mantendo a consistência", falei sobre a vantagem da valoração da proposição inteira antes de passar para a seguinte. Observe como fica a tabela depois que a primeira proposição é valorada:

P	Q	R	P	∧	Q	R	→	~	P	~	Q	↔	R
V	V	V	V	V	V								
V	V	F	V	V	V								
V	F	V	V	F	F	—	—	—	—	—	—	—	—
V	F	F	V	F	F	—	—	—	—	—	—	—	—
F	V	V	F	F	V	—	—	—	—	—	—	—	—
F	V	F	F	F	V	—	—	—	—	—	—	—	—
F	F	V	F	F	F	—	—	—	—	—	—	—	—
F	F	F	F	F	F	—	—	—	—	—	—	—	—

Quando você encontrar uma linha onde uma premissa for falsa ou a conclusão for verdadeira, desenhe traços até o final da linha. Isso servirá para lembrá-lo de não valorar as outras proposições dessas linhas.

A primeira coluna desse exemplo é especialmente útil, pois a primeira premissa é falsa em seis das oito linhas da tabela anterior, o que significa que você pode descartar essas seis linhas. Veja como fica o resto da tabela depois de preenchida:

P	Q	R	P	∧	Q	R	→	~	P	~	Q	↔	R
V	V	V	V	V	V	V	F	F	V	—	—	—	—
V	V	F	V	V	V	F	V	F	V	F	V	V	F
V	F	V	V	F	F	—	—	—	—	—	—	—	—
V	F	F	V	F	F	—	—	—	—	—	—	—	—
F	V	V	F	F	V	—	—	—	—	—	—	—	—
F	V	F	F	F	V	—	—	—	—	—	—	—	—
F	F	V	F	F	F	—	—	—	—	—	—	—	—
F	F	F	F	F	F	—	—	—	—	—	—	—	—

Quando nenhuma linha da tabela verdade contiver todas as premissas verdadeiras e uma conclusão falsa, o argumento será válido; caso contrário, ele será inválido.

Como você pode ver, a única linha na tabela anterior que tem todas as premissas verdadeiras também tem uma conclusão verdadeira, então o argumento é válido.

Juntando as Peças

Os tópicos anteriores deste capítulo mostram como usar a tabela verdade para testar uma variedade de condições lógicas. A Tabela 6-2 organiza essa informação.

Tabela 6-2: Testes de Tabela Verdade para Várias Condições Lógicas

Condição Sendo Testada	*Número de Proposições*	*Condição Confirmada Quando*
Tautologia	1	A proposição é verdadeira em todas as linhas.
Contradição	1	A proposição é falsa em todas as linhas.
Contingência	1	A proposição é verdadeira em pelo menos uma linha e falsa em pelo menos uma linha.
Equivalência	2	Ambas as proposições têm o mesmo valor Semântica verdade em cada uma das linhas.
Não equivalência	2	As duas proposições têm valores verdade diferentes em pelo menos uma linha.
Consistência	2 ou mais	Todas as proposições são verdadeiras em pelo menos uma linha.
Inconsistência	2 ou mais	Todas as proposições não são verdadeiras em todas as linhas.
Validade	2 ou mais	Em toda linha onde todas as

		premissas são verdadeiras, a conclusão também é verdadeira
Invalidade	2 ou mais	Todas as premissas são verdadeiras e a conclusão é falsa em ao menos uma linha.

Se você tem a impressão de que todos esses conceitos estão de algum modo conectados, está certo.

Conectando tautologias e contradições

Você pode facilmente transformar uma tautologia em contradição (ou viceversa), basta negar toda a proposição com um operador ~.

Lembre-se de que a proposição anterior

$$P \rightarrow (\sim Q \rightarrow (P \wedge \sim Q))$$

é uma tautologia. Então sua negação, a proposição

$$\sim(P \rightarrow (\sim Q \rightarrow (P \wedge \sim Q)))$$

é uma contradição. Para ter certeza disso, basta observar a tabela verdade para essa nova proposição:

P	Q	~	(P	→	(~Q	→	(P	∧	~Q)))
V	V	**F**	V	V	V	V	V	V	V
V	F	**F**	V	V	F	V	V	F	F
F	V	**F**	F	V	V	F	F	F	V
F	F	**F**	F	V	F	V	F	F	F

Como você pode ver, a única coisa que mudou é que o operador principal da proposição — o único operador fora dos parênteses —

é, agora, o operador ~.

Também deve estar claro que você pode transformar contradição em tautologia, da mesma maneira. Assim, a proposição

$$\sim(\sim(P \rightarrow (\sim Q \rightarrow (P \wedge \sim Q)))$$

é uma tautologia o que lhe mostra que, apesar de serem oposições, tautologias e contradições são muito parecidas

Relacionando equivalência semântica e tautologia

Quando você relaciona quaisquer duas proposições semanticamente equivalentes com o operador ↔, a proposição resultante é uma tautologia.

Como você viu anteriormente, as duas proposições

$$P \rightarrow Q \text{ e } \sim P \vee Q$$

são semanticamente equivalentes. Isto é, não importa quais sejam os valores lógicos de *P* e *Q*, as duas proposições terão a mesma valoração.

Agora relacione essas duas proposições com o operador ↔:

$$(P \rightarrow Q) \leftrightarrow (\sim P \vee Q)$$

O resultado é uma tautologia. Se duvida desse resultado, observe a tabela verdade para essa proposição:

P	Q	(P	→	Q)	↔	(~	P	∨	Q)
V	V	V	V	V	V	F	V	V	V
V	F	V	F	F	V	F	V	F	F
F	V	F	V	V	V	V	F	V	V
F	F	F	V	F	V	V	F	V	F

É claro que você pode transformar essa tautologia em uma contradição, negando-a, da seguinte forma:

$$\sim((P \rightarrow Q) \leftrightarrow (\sim P \vee Q))$$

Relacionando inconsistência e contradição

LEMBRE-SE

Quando você relaciona um conjunto de proposições inconsistentes com uma única proposição usando repetidamente o operador $\wedge$, a proposição resultante é uma contradição.

Como viu anteriormente, as três proposições

$$P \vee \sim Q$$

$$P \rightarrow Q$$

$$P \leftrightarrow \sim Q$$

são inconsistentes. Isto é, sob qualquer interpretação, ao menos uma delas é falsa.

Agora relacione essas três proposições usando operadores $\wedge$:

$$((P \vee \sim Q) \wedge (P \rightarrow Q)) \wedge (P \leftrightarrow \sim Q)$$

LEMBRE-SE

Quando você usa operadores para relacionar mais que duas proposições, precisa usar parênteses extras para que fique claro qual operador é o principal. Veja mais sobre isso no Capítulo 14.

O operador principal é o segundo operador $\wedge$ — o único operador que está fora dos parênteses —, mas, em qualquer hipótese, o resultado é uma contradição. Para confirmar esse resultado, você pode usar a tabela verdade para valorar a proposição em todas as interpretações. Primeiro, valore tudo que está dentro do primeiro conjunto de parênteses:

P	Q	((P	∨	~	Q)	∧	(P	→	Q))	∧	(P	↔	~	Q)
V	V	V	$\underline{V}$	F	V	V	V	$\underline{V}$	V		V	F	F	V
V	F	V	$\underline{V}$	V	F	F	V	$\underline{F}$	F		V	V	V	F
F	V	F	$\underline{F}$	F	V	F	F	$\underline{V}$	V		F	V	F	V
F	F	F	$\underline{V}$	V	F	V	F	$\underline{V}$	F		F	F	V	F

Depois, valore a proposição inteira:

P	Q	((P	∨	~	Q)	∧	(P	→	Q))	∧	(P	↔	~	Q)
V	V	V	V	F	V	$\underline{V}$	V	V	V	F	V	$\underline{F}$	F	V
V	F	V	V	V	F	$\underline{F}$	V	F	F	F	V	$\underline{V}$	V	F
F	V	F	F	F	V	$\underline{F}$	F	V	V	F	F	$\underline{V}$	F	V
F	F	F	V	V	F	$\underline{V}$	F	V	F	F	F	$\underline{F}$	V	F

Conforme previsto, a proposição é valorada como falsa em todas as linhas da tabela, então é uma contradição.

Relacionando validade e contradição

Como você já deve ter adivinhado, a validade do argumento também pode ser entrelaçada nessa rede. Veja meu exemplo favorito de argumento válido (sim, pessoas que gostam de Lógica têm seus favoritos):

Premissas:

$P \rightarrow Q$

$Q \rightarrow R$

Conclusão:

$P \rightarrow R$

Por esse ser um argumento válido, você sabe que é impossível que ambas as premissas sejam verdadeiras e a conclusão, falsa. Em outras palavras, se você construir uma tabela verdade, nenhuma das linhas poderia ficar assim:

$P \to Q$	$Q \to R$	$P \to R$
V	V	F

Da mesma forma, se negar a conclusão, usando um operador ~ e, então, construir outra tabela verdade, *nenhuma* linha poderia ficar assim:

$P \to Q$	$Q \to R$	$\sim P \to R$
V	V	V

Mas, se *nenhuma* interpretação fizer com que todas essas proposições sejam verdadeiras, você pode considerar que esse é um conjunto de proposições inconsistentes.

Fazendo ainda mais conexões

Para os puristas entre nós que precisam saber como tudo se encaixa:

✓ Quando você relacionar quaisquer duas proposições semanticamente não equivalentes com o operador $\leftrightarrow$, a proposição resultante *não será* uma tautologia, isto é, ela será uma contingência ou uma contradição.

✓ Quando você negar a conclusão de um argumento inválido, obterá um conjunto de proposições consistente.

✓ Quando você relacionar um conjunto de proposições inconsistentes com uma única proposição usando o operador ^ repetidamente, a proposição resultante *não* será uma contradição, isto é, ela será uma contingência ou uma tautologia.

Quando você nega a conclusão de um argumento válido, obtém um conjunto de proposições inconsistentes (pode parecer contraditório que validade e inconsistência estejam relacionadas, mas é assim que funciona). Você ainda pode transformar esse conjunto de proposições inconsistentes em uma contradição, relacionando-as com o operador ^:

((P	→	Q)	&	(Q	→	R))	∧	~	(P	→	R)
							V				

Para transformar um argumento válido em uma proposição contraditória, relacione todas as premissas com a negação da conclusão, repetindo o operador ^.

Capítulo 7

Escolhendo o Caminho Mais Fácil: Criando Tabelas Rápidas

Neste Capítulo

- Procurando os valores verdade das proposições inteiras
- Entendendo como construir, preencher e ler uma tabela rápida
- Sabendo com que tipo de proposição você está trabalhando

Você já leu alguns capítulos deste livro e já deve estar pegando o jeito de como trabalhar com as tabelas verdade e ficando bom nisso — talvez até um pouco confiante demais.

Por exemplo, suponha que segunda-feira bem cedo, seu professor entre na classe com uma xícara de café, duas rosquinhas e o jornal, e mande que você escreva a tabela verdade para a seguinte proposição:

$$P \rightarrow ((Q \wedge R) \vee (\sim P \wedge S))$$

Aí ele senta, abre o jornal e ignora todos.

Com quatro constantes, você está falando de dezesseis linhas de purgatório de valoração. Mas você é um encrenqueiro típico e, como tal, vai até a mesa dele e diz que o operador principal da proposição é →, pois ele é o único operador fora dos parênteses. Ele sacode o jornal irritado.

Você insiste cuidadosamente e explica a clareza de sua nova descoberta: “Você não vê? Todas as oito interpretações que partem de *P* como falsas têm que fazer com que a proposição inteira seja verdadeira. Não está certo?”. Ele dá uma mordida na rosquinha e usa isso como desculpa para não dizer nada enquanto o encara.

Finalmente, você toma coragem e pergunta: “Então, posso marcar todas essas oito linhas como verdadeiras, sem ter que fazer todos os passos?”.

Ele o censura: “Não!”, o que faz com que você volte de fininho para sua mesa.

Como já disse, você é um agitador. Provavelmente é daquele tipo que lê (ou até escreve) os livros *Para Leigos.* Continue lendo, sabichão, para ver no que isso vai dar.

Você não está tão distante da realidade com seu modo de pensar. Há um método melhor do que o de “preencher e encaixar” para problemas como esse (a menos que seu professor seja especialmente cruel) — a *tabela rápida*! Ao contrário das tabelas verdade, que requerem a avaliação do problema sob *cada* uma das possíveis interpretações, as tabelas rápidas usam apenas uma linha para fazer todo o trabalho da anterior.

Neste capítulo, você verá como as tabelas rápidas economizam tempo, trabalhando com proposições como um *todo*, ao invés de em *partes*, como as tabelas verdade fazem. Mostrarei como reconhecer os tipos de problemas que são mais facilmente resolvidos com as tabelas rápidas do que com as tabelas verdade. Demonstrarei ainda as estratégias e métodos para ajudá-lo a usar as tabelas rápidas para resolver uma variedade de problemas comuns.

Trocando a Tabela Verdade por uma Nova Amiga: A Tabela Rápida

As tabelas verdade são ordenadas, precisas, completas — e entediantes! Isso porque você tem que valorar *cada* interpretação possível para resolver um problema.

Quando usa as tabelas verdade, você começa pelas *partes* da proposição (os valores verdade das constantes) e termina pela proposição *inteira* (o valor do operador principal). Esse método de começar pelas partes e terminar pelo todo é tanto a força quanto a fraqueza desse tipo de tabela. Considerando que você tem que avaliar a proposição sob cada interpretação possível, você precisa estar certo de que cobriu todas as possibilidades. Pela mesma razão, porém, muito do trabalho é repetitivo, em outras palavras, entediante e cansativo!

Mas como você já viu acontecer com o encrenqueiro do começo do capítulo, muitas dessas interpretações tendem a ser redundantes. Em consequência, em muitos casos você pode eliminar grande parte delas de uma só vez. Ao mesmo tempo, porém, você precisa ter cuidado para desprezar apenas as interpretações erradas e guardar as certas. Para ter certeza de que está descartando aquelas que são adequadas, precisa de um sistema (e isso é o que este capítulo vai lhe mostrar).

Ao contrário do que acontece com as tabelas verdade, nas rápidas você começa com a proposição *inteira* — o valor lógico do operador principal — e termina com os valores das *partes*, que são os valores das constantes. A ideia por trás desse método é começar com apenas um valor verdade para a proposição e, tomando algumas decisões inteligentes, economizar um tempo enorme eliminando o trabalho repetitivo.

Você pode usar as tabelas rápidas para substituir as tabelas verdade a fim de testar quaisquer das condições discutidas no Capítulo 6 e listadas na Tabela 6-2.

DICA

Falando de forma geral, os três tipos de problemas a seguir são daqueles em que podemos desprezar a velha tabela verdade e usar as tabelas rápidas:

- ✓ **Problemas que seu professor manda serem resolvidos com o uso das tabelas rápidas**: Não precisa nem explicar!
- ✓ **Problemas com quatro ou mais constantes:** Tabelas grandes significam problemas grandes. É muito provável que você cometa erros, não importa o quanto seja cuidadoso. Nesses casos, as tabelas rápidas são a garantia de que, na prática, você irá economizar tempo.
- ✓ **Problemas com proposições que são dos "tipos fáceis":** Alguns tipos de proposições são fáceis de resolver com as tabelas rápidas. Mostrarei como reconhecer esses tipos mais adiante, neste capítulo, no tópico "Trabalhando Melhor (Não Mais) com as Tabelas Rápidas".

Descrevendo o Processo das Tabelas Rápidas

Neste tópico, apresentarei uma visão geral dos três passos básicos para usar as tabelas rápidas. Exemplos são provavelmente a melhor maneira de se entender como usar essas tabelas. Então explicarei com um exemplo e depois mostrarei os detalhes nos tópicos seguintes.

Cada um dos três passos para usar a tabela rápida pode ser ainda mais fácil se você souber alguns truques de como abordar a tabela. Esses truques serão apresentados um pouco mais adiante. Continue acompanhando e, se tiver alguma dúvida, encontrará as respostas mais adiante neste capítulo.

Eis o exemplo que você acompanhará nos próximos tópicos: suponha que você queira saber se as seguintes proposições são consistentes ou inconsistentes:

$$P \wedge Q \qquad Q \rightarrow R \qquad R \rightarrow P$$

Fazendo uma suposição estratégica

Todas as tabelas rápidas começam com uma *suposição estratégica.* Nesse exemplo, suponha que todas as três proposições sejam verdadeiras e veja onde isso o leva.

Considerando essa suposição estratégica, toda tabela rápida pode levar a dois resultados possíveis:

> ✓ Encontrar uma interpretação sob sua suposição
>
> ✓ Refutar a suposição mostrando que não existe essa interpretação

Dependendo do problema que você está resolvendo, cada resultado o leva a uma diferente conclusão.

Pense no problema do exemplo da seguinte forma: se as proposições *P* ^ *Q, Q* → *R* e *R* → *P* são inconsistentes, os valores verdade de todas as três proposições são **V** sob pelo menos uma das interpretações (veja o Capítulo 6 para relembrar a definição de consistência).

Dessa forma, uma boa estratégia é assumir que o valor verdade de cada proposição é **V** e, com isso, ver se pode fazer essa suposição funcionar. Por exemplo, assim é como sua tabela vai ficar:

$P \wedge Q$	$Q \rightarrow R$	$R \rightarrow P$
V	V	V

Dê uma olhadinha no tópico “Planejando Sua Estratégia”, mais adiante neste capítulo, para ver como montar uma tabela rápida e resolver cada tipo de problema que já resolveu com a tabela verdade. Lá, você encontrará uma lista completa das presunções estratégicas que deve usar para cada tipo de problema.

Preenchendo uma tabela verdade

Depois de montar a tabela rápida, procure quaisquer outras conclusões que possam ser formuladas sobre os valores verdade de qualquer das partes da proposição. No caso do exemplo que está sendo utilizado, se a proposição *P* ^ *Q* for verdadeira, ambas as subproposições *P* e *Q* também são verdadeiras:

$P \wedge Q$	$Q \rightarrow R$	$R \rightarrow P$
V V V	V	V

Depois de saber que *P* e *Q* são verdadeiras, você pode inserir essa informação em todos os lugares em que as constantes aparecem assim:

$P \wedge Q$	$Q \rightarrow R$	$R \rightarrow P$
V V V	V V	V V

Agora observe a segunda proposição, $Q \rightarrow R$. A proposição inteira é verdadeira e a primeira parte também o é, então a segunda parte também é verdadeira. Portanto, R é verdadeira em todos os lugares em que aparece. Então, nossa tabela deve ficar assim:

$P \wedge Q$	$Q \rightarrow R$	$R \rightarrow P$
V V V	V V V	V V V

Nesse ponto, todas as constantes já foram preenchidas e você está pronto para ler sua tabela verdade.

Lendo uma tabela rápida

Quando você termina de preencher a tabela rápida, tem uma interpretação *possível*, mas precisa ter certeza de que ela realmente funciona.

Quando você imaginar que chegou a uma interpretação correta, certifique-se de que:

- ✓ Cada constante tenha o mesmo valor verdade em todas as posições em que aparecer.
- ✓ Cada valoração esteja correta para aquela interpretação.

O exemplo que está sendo usado passa em ambos os testes. Cada uma das três variáveis tem o mesmo valor verdade em todas as posições (por exemplo, o valor de P é sempre **V**). Cada valoração está correta sob essa interpretação (por exemplo, o valor de $P \wedge Q$ é **V**, o que está correto).

Assim, você encontrou uma interpretação que torna a suposição original correta, o que significa que todas as três proposições são consistentes.

Refutando a suposição

No exemplo sob análise, a suposição leva a uma interpretação. Mas, conforme foi dito no tópico “Fazendo uma suposição estratégica”, deste capítulo, isso não acontece sempre. Algumas vezes você pode achar uma suposição que o leve a uma situação impossível.

Por exemplo, suponha que você queira saber se a proposição (*P* ^ *Q*) ^ ((*Q* ↔ *R*) ^ ~*P*) é uma contradição, isto é, se o seu valor verdade é **F** sob qualquer interpretação.[1]

Como sempre, você começa com uma suposição estratégica. Nesse caso, assuma que a proposição *não é* uma contradição e, assim, seu valor verdade é **V** sob pelo menos uma interpretação:

$$\begin{array}{ccccccccc} (P & \land & Q) & \land & ((Q & \leftrightarrow & R) & \land & \sim P) \\ & & & V & & & & & \end{array}$$

Como já visto no Capítulo 5, o operador principal dessa proposição — o único operador fora de todos os parênteses — é o segundo operador ^. Com isso, ela tem a forma de *x* ^ *y*. Supondo que a proposição inteira seja verdadeira, você pode concluir que ambas as subproposições também são verdadeiras, o que deixaria sua tabela assim:

$$\begin{array}{ccccccccc} (P & \land & Q) & \land & ((Q & \leftrightarrow & R) & \land & \sim P) \\ & V & & V & & & & V & \end{array}$$

Porém, note que ambas as proposições *P* ^ *Q* e *(Q* ↔ *R)* ^ ~*P* também estão na forma *x* ^ *y*, o que significa que *P, Q, Q* ↔ *R* e ~*P* são todas verdadeiras:

$$\begin{array}{ccccccccc} (P & \land & Q) & \land & ((Q & \leftrightarrow & R) & \land & \sim P) \\ V & V & V & V & & V & & V & V \end{array}$$

Por enquanto, tudo bem. Você já fez muito progresso em apenas dois passos. No entanto, agora vem o problema: parece que *P êP* são ambas verdadeiras, e isso obviamente não pode estar correto.

Essa impossibilidade refuta a suposição original de que essa proposição *não é* uma contradição. Portanto, a proposição *é* uma contradição.

---

1 N.E.: O tipo de prova neste exemplo denomina-se “prova por absurdo”. Começa-se supondo um resultado oposto do que você quer provar e chega-se, a partir daí, a uma afirmação impossível que neste caso foi o dos valores lógicos *P* e ~*P* serem ambos verdade.

Quando você refuta uma suposição, precisa ter certeza de que descartou todas as interpretações possíveis. O motivo que o levou a descartar essas interpretações não fica evidenciado na tabela rápida já preenchida, o que faz com que alguns professores possam pedir uma breve explicação de como você chegou à conclusão de que deveria descartá-las.

Este é o tipo de explicação que serviria perfeitamente para justificar o exemplo demonstrado: “suponha que a proposição *não é* uma contradição. Então, pelo menos, uma das interpretações torna a proposição verdadeira. Assim, $(P \wedge Q)$ e $((Q \leftrightarrow R) \wedge \sim P)$ são verdadeiras. Mas, tanto *P* quanto ~*P* são verdadeiras, o que é impossível, de forma que a proposição é uma contradição”.

Planejando Sua Estratégia

Quando testa quaisquer das condições listadas no Capítulo 6, Tabela 6-2, você começa com uma suposição estratégica e, depois, procura por uma interpretação que se encaixe. Se você realmente achar essa interpretação, obterá uma resposta, mas, se descobrir que ela não existe, obterá outra resposta.

Este tópico é um resumo da informação de que você precisa para montar e ler uma tabela rápida. Darei, para cada caso, a suposição de que você precisa para começar, um exemplo e seu primeiro passo e, então, mostrarei como ler ambos os resultados possíveis.

Em cada caso, a suposição estratégica que lhe darei é a melhor forma (e às vezes a única) de testar determinada condição usando a tabela rápida. Isso porque cada suposição permite que você formule uma conclusão baseada na existência (ou não) de uma *única* interpretação. As tabelas rápidas são feitas especialmente para encontrar uma única interpretação, se houver alguma.

Tautologia

Suposição estratégica: tente demonstrar que a proposição não é uma tautologia, presumindo que ela é falsa.

Exemplo: $((P \rightarrow Q \rightarrow R)) \rightarrow ((P \wedge Q) \rightarrow R)$ é uma tautologia?

Primeiro passo:

$$\begin{array}{c} (P \rightarrow Q \rightarrow R) \rightarrow ((P \wedge Q) \rightarrow R) \\ \mathrm{F} \end{array}$$

Resultados:

> ✓ **Se você encontrar uma interpretação sob essa suposição:** A proposição *não é* uma tautologia – é uma contradição ou uma contingência.
>
> ✓ **Se você refutar a suposição:** A proposição é uma tautologia.

Contradição

Suposição estratégica: tente demonstrar que a proposição não é uma contradição, presumindo que ela é verdadeira.

Exemplo: ((*P* → *Q* → *R*)) → ((*P* ∧ *Q*) → *R*) é uma contradição?

Primeiro passo:

$$(P \rightarrow Q \rightarrow R) \underset{V}{\rightarrow} ((P \wedge Q) \rightarrow R)$$

Resultados:

> ✓ **Se você encontrar uma interpretação sob essa suposição:** A proposição *não for* uma contradição – é uma tautologia ou uma contingência.
>
> ✓ **Se você refutar a suposição:** A proposição é uma contradição.

Contingência

Use os dois testes anteriores para tautologia e contradição. Se a proposição *não for* uma tautologia *nem* uma contradição, deve ser uma contingência.

Equivalência semântica e não equivalência

Suposição estratégica: tente demonstrar que duas proposições são semanticamente não equivalentes, conectando-as por meio do operador ↔, e assuma que essa nova proposição é falsa.

Exemplo: $P \wedge (Q \vee R)$ e $(P \vee Q) \wedge (P \vee R)$ são proposições semanticamente equivalentes?

Primeiro passo:

$$\begin{array}{c} (P \wedge (Q \vee R)) \leftrightarrow ((P \vee Q) \wedge (P \vee R)) \\ \mathrm{F} \end{array}$$

Resultados:

> ✓ **Se você encontrar uma interpretação sob essa suposição:** As proposições são semanticamente não equivalentes.
>
> ✓ **Se você refutar a suposição:** As proposições são semanticamente equivalentes.

Consistência e inconsistência

Suposição estratégica: tente mostrar que o conjunto de proposições é consistente, assumindo que todas elas são verdadeiras.

Exemplo: As proposições $P \wedge Q$, $\sim(\sim Q \vee R)$ e $\sim R \rightarrow \sim P$ são consistentes ou inconsistentes?

Primeiro passo:

$$\begin{array}{ccc} P \wedge Q & \sim(\sim Q \vee R) & \sim R \rightarrow \sim P \\ \mathrm{V} & \mathrm{V} & \mathrm{V} \end{array}$$

Resultados:

> ✓ **Se você encontrar uma interpretação sob essa suposição:** O conjunto de proposições é consistente.
>
> ✓ **Se você refutar a suposição:** O conjunto de proposições é inconsistente.

Validade e invalidade

Suposição estratégica: tente demonstrar que o argumento é inválido, assumindo que todas as premissas são verdadeiras e que a conclusão é falsa.

Exemplo: o argumento é válido ou inválido?

Premissas:

$P \to Q$

$\sim(P \leftrightarrow R)$

Conclusão:

$\sim(\sim Q \wedge R)$

Primeiro passo:

$P \to Q$	$\sim(P \leftrightarrow R)$	$\sim(\sim Q \wedge R)$
V	V	F

Resultados:

> ✓ **Se você encontrar uma interpretação sob essa suposição:** O argumento é inválido.
>
> ✓ **Se você refutar a suposição:** O argumento é válido.

Trabalhando Melhor (e Não Mais) com as Tabelas Rápidas

Parte do dilema de usar tabelas rápidas é que você tem que pensar em como proceder, ao invés de simplesmente escrever todas as possibilidades. Dessa forma, as tabelas rápidas se tornam muito mais fáceis quando você sabe o que está procurando. Neste tópico, mostrarei como usar tabelas rápidas para obter melhores resultados.

No Capítulo 5, foram vistas as oito formas básicas de proposições de LS como um modo de entender a valoração. Essas formas são ainda mais úteis quando você usa as tabelas rápidas; dê uma olhadinha na Tabela 5-1, se precisar de uma revisão rápida.

Quando você está trabalhando com tabelas rápidas, os valores verdade de cada forma básica de proposição se tornam importantes. Dois possíveis valores verdade (**V** e **F**) para cada uma das oito formas resultam em 16 diferentes possibilidades. Alguns desses tipos são mais fáceis de usar nas tabelas rápidas do que outros. Começarei pelos mais fáceis.

Reconhecendo os seis tipos mais fáceis de proposições para trabalhar

Dos 16 tipos de proposições de LS (incluindo os valores verdade), 6 são fáceis de trabalhar usando as tabelas rápidas. Em cada um desses tipos, os valores verdade das duas subproposições, *x* e *y*, são simples de descobrir.

Por exemplo, suponha uma proposição na forma $x \wedge y$, e que você saiba que o seu valor verdade é **V**. Lembre-se de que a única forma de uma proposição $\wedge$ ser verdadeira é quando ambas as suas partes também são, de modo que você sabe que o valor de ambas, *x* e *y*, também são **V**.

Da mesma forma, suponha uma proposição na forma ~(*x* ∧ *y*), e que você saiba que seu valor verdade é **F**. Nesse caso, é fácil ver que o valor de *x* ∧ *y* é **V**, o que novamente significa que os valores de *x* e *y* são **V**.

A Figura 7-1 mostra os seis tipos de proposições mais fáceis de se trabalhar na LS. Depois que você reconhecer essas proposições, poderá se mover rapidamente na tabelas rápidas.

Começando com:	Leva a:	Valores de x e y:
x ∧ y OU ~(x ∧ y) V F	x ∧ y V v V	x é V e y é V
x ∨ y OU ~(x ∨ y) F V	x ∨ y F F F	x é F e y é F
x → y OU ~(x → y) F V	x → y V F F	x é V e y é F

Figura 7-1: Os seis tipos mais fáceis de proposições na LS.

Por exemplo, suponha que você queira descobrir se o seguinte argumento é válido ou inválido:

Premissas:

~(*P* → (*Q* ∨ *R*))

~(*P* ∧ (*Q* ↔ ~*R*))

Conclusão:

(*P* ∧ ~*R*)

O primeiro passo é sempre escolher a estratégia correta. Nesse caso, como demonstro no tópico “Planejando Sua Estratégia”, anteriormente neste capítulo, você assume que as premissas são verdadeiras e que a conclusão é falsa. Em outras palavras, supõe que o argumento é *inválido* e procura uma interpretação que se encaixe nessa suposição. Sua tabela ficará assim:

~(P → (Q ∨ R))	~(P ∧ (Q ↔ ~R))	(P ∧ ~R)
V	V	F

Observe que a primeira proposição está na forma ~(*x* → *y*), com um valor verdade **V.** Olhando a Figura 7-1, você sabe que *P* é verdadeira e *Q* ᵛ *R* é falsa, o que significa que pode preencher sua tabela assim:

~(P → (Q ∨ R))	~(P ∧ (Q ↔ ~R))	(P ∧ ~R)
V V F F	V	F

Agora que sabe que o valor de *Q* ᵛ *R* é **F**, olhando novamente a Figura 7-1, você vê que *Q* e *R* são ambas falsas:

~(P → (Q ∨ R))	~(P ∧ (Q ↔ ~R))	(P ∧ ~R)
V V F F F F	V	F

Em apenas três passos, você descobriu os valores verdade de todas as três constantes, e pode preencher o restante da tabela da seguinte forma:

~(P → (Q ∨ R))	~(P ∧ (Q ↔ ~R))	(P ∧ ~R)
V V F F F F	V V F F	V F F

Agora você precisa terminar de preencher a tabela e verificar se a interpretação funciona para todas as proposições:

~(P → (Q ∨ R))	~(P ∧ (Q ↔ ~R))	(P ∧ ~R)
V V F F F F	V V F F F V F	V F V F

Nesse caso, a segunda proposição está correta. No entanto, a terceira está incorreta: ambas as partes da proposição ^ são verdadeiras, de forma que o valor verdade da proposição inteira não pode ser **F**. Isso refuta a suposição de que o argumento é *inválido* e significa que o argumento é *válido.*

Trabalhando com os quatro tipos de proposição não tão fáceis

Algumas vezes você fica empacado com alguns tipos de proposição da LS que não são tão fáceis de trabalhar, como as seis primeiras que apresentei no tópico anterior. Isso é especialmente verdade quando você está testando a equivalência semântica, pois, conforme já visto anteriormente neste capítulo, no tópico “Planejando Sua Estratégia”, a estratégia aqui é juntar duas proposições com o operador ↔.

Todos os quatro tipos de proposição que contêm o operador ↔ lhe dão *dois* possíveis conjuntos de valores para *x* e *y*, conforme demonstrado na Figura 7-2.

Começando com:	Leva a:	Valores de x e y:
x ↔ y OU ~(x ↔ y) V F	x ↔ y V v V F v F	x é V e y é V OU x é F e y é F
x ↔ y OU ~(x ↔ y) F V	x ↔ y V F F F F V	x é V e y é F OU x é F e y é V

Figura 7-2: Quatro tipos não tão fáceis de proposições na LS.

Suponha que você queira descobrir se as proposições ~(*P* ∨ (*Q* → *R*)) e ((*P* → *R*) ∧ *Q*) são semanticamente equivalentes ou não. A estratégia aqui é relacionar as duas proposições com o operador ↔ e, depois, assumir que essa nova proposição é falsa. Em outras palavras, assuma que as duas proposições são semanticamente não equivalentes. Sua tabela deve ficar assim:

$$\begin{array}{ccc} \sim(P \vee (Q \to R)) & \leftrightarrow & ((P \to R) \wedge Q) \\ & F & \end{array}$$

Como a Figura 7-2 mostra, você pode tomar dois caminhos possíveis para essa proposição:

$$\begin{array}{lcr} \sim(P \vee (Q \to R)) & \leftrightarrow & ((P \to R) \wedge Q) \\ V & F & F \\ F & F & V \end{array}$$

Quando olha para o primeiro caminho, você observa que a primeira parte da proposição é um dos seis tipos que são fáceis de trabalhar, de modo que pode preencher sua tabela da seguinte forma:

```
~(P ∨ (Q → R)) ↔ ((P → R) ∧ Q)
V F F    F     F          F
```

Além disso, sabendo que *Q* → *R* é falsa, você pode concluir que *Q* é verdadeira e *R* é falsa e, agora, pode preencher sua tabela assim:

```
~(P ∨ (Q → R)) ↔ ((P → R) ∧ Q)
V F F  V F F   F          F
```

Depois de saber os valores de todas as três constantes, você pode preencher o resto da tabela:

```
~(P ∨ (Q → R)) ↔ ((P → R) ∧ Q)
V F F  V F F   F   F V F  F V
```

Sob essa interpretação, as duas partes da proposição ^ são verdadeiras, mas a proposição em si é falsa, o que está incorreto. Então, a busca por uma interpretação que funcione continua:

Agora você pode tentar o segundo caminho:

```
~(P ∨ (Q → R)) ↔ ((P → R) ∧ Q)
F              F          V
```

Por este caminho, a segunda parte da proposição é um tipo fácil, o que permite que a tabela seja preenchida assim:

```
~(P ∨ (Q → R)) ↔ ((P → R) ∧ Q)
F              F     V    V V
```

Agora você sabe que *Q* é verdadeira. A proposição ᵛ, na primeira parte, também é verdadeira, pois sua negação é falsa. Então, preencha a tabela da seguinte forma:

```
~(P ∨ (Q → R)) ↔ ((P → R) ∧ Q)
F   V  V       F     V    V V
```

Nesse ponto você não tem mais conclusões sólidas para elaborar. Mas está próximo do fim e, com a tabela rápida, precisa encontrar apenas uma interpretação que funcione. Nesse caso, sugiro que você levante uma hipótese e veja onde ela o leva.

Suponha, por exemplo, que o valor de *P* seja **V**. Assim, para fazer com que a subproposição *P* → *R* seja verdadeira, o valor de *R* também teria que ser **V**. Então, parece que você obteve uma interpretação perfeita da proposição, que ocorre quando o valor das três constantes é **V**. Preencha a tabela até que fique assim:

~	(P	∨	(Q	→	R))	↔	((P	→	R)	∧	Q)
F	**V**	V	V	**V**	**V**	F	**V**	V	**V**	V	V

Essa hipótese se confirma, pois, conforme já dito anteriormente, no tópico “Lendo a tabela rápida”, todas as constantes têm o mesmo valor verdade em todas as posições, e a valoração da proposição inteira, sob essa interpretação, está correta. Então você encontrou uma interpretação sob sua suposição original, o que significa que as duas proposições são semanticamente não equivalentes.

Você pode estar imaginando o que teria acontecido se tivesse assumido que o valor de *P* era **F**. Nesse caso, você teria encontrado uma interpretação alternativa, com o valor lógico de *R* sendo **V**. Mas isso não importa, pois com as tabelas rápidas, você só precisa encontrar uma interpretação e pronto.

Enfrentando os seis tipos de proposições difíceis

Seis tipos de proposições da LS não se adaptam muito bem às tabelas rápidas. A Figura 7-3 mostra por que isso acontece.

Começando com:	Leva a:	Valores de x e y:
x ∧ y OU ~(x ∧ y) F V	x & y V F F F F V F F F	x é V e y é F OU x é F e y é V OU x é F e y é F
x ∨ y OU ~(x ∨ y) V F	x ∨ y V v V V v F F v V	x é V e y é V OU x é V e y é F OU x é F e y é V
x → y OU ~(x → y) V F	x → y V v V F v V F v F	x é V e y é V OU x é F e y é V OU x é F e y é F

Figura 7-3: Os seis tipos difíceis de proposições da LS.

Como você pode ver, cada um desses tipos de proposições o leva a três possíveis caminhos e, se você está usando uma tabela rápida para resolver esse problema, entrará em uma busca longa e enfadonha. Felizmente há outras opções. Os tópicos seguintes mostram quais serão suas opções quando se deparar com três resultados possíveis.

O primeiro caminho: Use a tabela verdade

Uma maneira de evitar a complicação desses tipos de proposições é usar as tabelas verdade. Elas são sempre uma opção (a menos que seu professor o proíba de utilizá-las) e sempre lhe dão a resposta correta. O lado negativo das tabelas verdade, como sempre, é que se o problema tem muitas constantes diferentes, você terá muito trabalho.

O segundo caminho: Use a tabela rápida

O segundo modo de resolver um problema difícil é aguentar firme e encarálo, experimentando os três caminhos com a tabela rápida. Afinal, você já descobriu como experimentar dois caminhos no exemplo anterior. Além disso, pode ter a sorte de encontrar uma interpretação que funcione na primeira tentativa.

Por exemplo, suponha que você queira descobrir se a proposição (~(*P* → *Q*) ∧ ~(*R* ∨ *S*) → (*Q* ↔ *R*) é uma contradição. A estratégia neste caso é fazer a seguinte suposição:

```
(~(P → Q) ∧ ~(R ∨ S)) → (Q ↔ R)
                      V
```

o que leva a três caminhos possíveis:

```
(~(P → Q) ∧ ~(R ∨ S)) → (Q ↔ R)
          V           V    V
          F           V    V
          F           V    F
```

Felizmente, o primeiro caminho o leva a um lugar promissor: observe que a subproposição (~(*P* → *Q*) ∧ ~(*R* ∨ *S*)) é uma proposição ∧, cujo valor é **V**, um dos seis tipos mais fáceis de proposição. A tabela, então, ficará assim:

```
(~(P → Q) ∧ ~(R ∨ S)) → (Q ↔ R)
 V        V V         V    V
```

Melhor ainda, as duas proposições menores também são tipos mais fáceis, o que deixa sua tabela assim:

```
(~(P → Q) ∧ ~(R ∨ S)) → (Q ↔ R)
 V V F F  V V F F F   V    V
```

Agora você pode inserir os valores de *Q* e *R:*

```
(~(P → Q) ∧ ~(R ∨ S)) → (Q ↔ R)
 V V F F  V V F F F   V  F V F
```

Em apenas três passos, você encontrou uma interpretação que funciona sob a suposição de que a proposição é verdadeira, então, ela não é uma contradição.

Nem sempre você terá essa sorte, mas pode ver que, nesse caso, a tabela rápida ainda foi uma opção muito mais rápida do que a tabela verdade. Boa sorte!

O terceiro caminho: Use a árvore lógica

No Capítulo 8, você aprenderá que as *árvores lógicas* são a alternativa perfeita para situações como essa. Quando as tabelas verdade são muito longas e as tabelas rápidas muito complicadas, a árvore lógica é sua melhor amiga.

Capítulo 8
A Verdade Nasce em Árvores

Neste Capítulo

- Decompondo proposições da LS com árvores lógicas
- Testando a consistência, a validade e a equivalência semântica
- Classificando tautologias, contradições e contingências com as árvores lógicas

Nos Capítulos 6 e 7, mostrei como usar as tabelas verdade e rápida para obter informações das proposições da Lógica Sentencial (LS) e de conjuntos de proposições. Neste capítulo, apresentarei um terceiro (e meu favorito!) método para resolver problemas lógicos: as árvores lógicas.

Nos tópicos a seguir, você verá como as árvores lógicas funcionam para decompor proposições em suas subproposições na LS. Assim, aprenderá como utilizá-las para resolver os mesmos tipos de problemas que antes resolvia com as tabelas verdade e rápida.

Entendendo o Funcionamento das Árvores Lógicas

Árvores lógicas são um poderoso instrumento sob qualquer aspecto. De fato, acho que elas são a melhor ferramenta deste livro para resolver quase todos os tipos de problemas encontrados na Lógica. Por quê? Fico feliz que tenha perguntado.

Em primeiro lugar, as árvores lógicas são fáceis: de aprender e de usar.

Árvores lógicas combinam os melhores aspectos de ambas as tabelas, a verdade e a rápida, mas sem seus problemas. Por exemplo, como as tabelas verdade, as árvores lógicas são métodos de “preencher e interpretar” (embora algumas escolhas inteligentes ao longo do caminho possam ser de grande ajuda), mas são muito mais curtos. Da mesma forma que as tabelas rápidas, as árvores lógicas evitam a valoração repetitiva, com a diferença de que não o levam a problemas cheios de conjecturas ou com três caminhos possíveis a serem descartados.

Elas são o caminho perfeito para resolver problemas da LS, de *qualquer* tamanho e com a máxima eficiência. São ainda muito úteis na Lógica Quantitativa (LQ), o maior sistema lógico, que será visto na Parte IV.

No Capítulo 5, discuti as oito formas básicas de proposições LS. Mais uma vez elas serão importantes para você entender as árvores lógicas, pois estas lidam com essas formas de uma maneira diferente. (Se você precisar de uma revisão dessas oito formas dê uma olhada na Tabela 5-1.)

Decompondo as proposições LS

As árvores lógicas funcionam *decompondo* proposições, isto é, quebrando-as em subproposições menores.

Por exemplo, se você sabe que a proposição inteira $(P \vee Q) \wedge (Q \vee R)$ é verdadeira, sabe que ambas as subproposições $(P \vee Q)$ e $(Q \vee R)$ também o são. De modo geral, você pode quebrar qualquer proposição verdadeira no formato $x \wedge y$ em duas proposições verdadeiras no formato x e y.

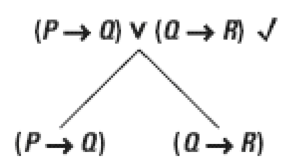
Em alguns casos, decompor a proposição significa dividi-la em duas, onde ao menos uma delas é verdadeira. Por exemplo, se você sabe que $(P \rightarrow Q) \vee (Q \leftrightarrow R)$ é verdadeira, sabe que tanto a subproposição $(P \rightarrow Q)$ quanto a $(Q \leftrightarrow R)$ também o são. De um modo geral, você pode quebrar qualquer proposição verdadeira no formato $x \vee y$ em duas, na forma x e y, onde ao menos uma delas será verdadeira.

LEMBRE-SE

As formas das proposições verdadeiras que o levam diretamente a outras são chamadas de *ramo único.* Já aquelas que o levam a duas possíveis direções são chamadas de *ramo duplo*. Essas sempre o levam a uma ou duas subproposições por ramo.

A Figura 8-1 mostra uma lista das oito formas proposicionais básicas de LS e suas decomposições.

Por exemplo, para decompor a proposição $(P \rightarrow Q) \vee (Q \rightarrow R)$, você começa por separá-la em duas subproposições, da seguinte forma:



Observe que assinalei a proposição $(P \rightarrow Q) \vee (Q \rightarrow R)$, depois que a dividi em duas subproposições. O sinal ao lado informa que o trabalho com ela está terminado, embora agora seja preciso trabalhar com as outras duas resultantes.

Em seguida, você divide cada uma das subproposições em outras ainda menores, de acordo com as regras mostradas na Figura 8-1.

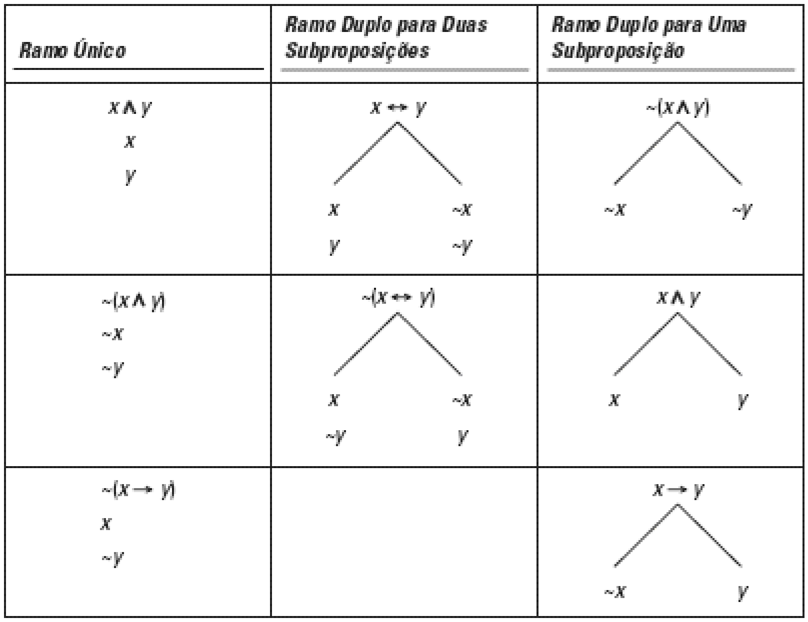
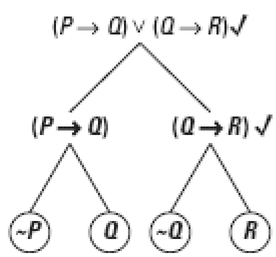


Figura 8-1: Os oito tipos de proposições da LS e suas decomposições.

Você pode ver agora de onde surgiu o nome das árvores lógicas. A estrutura final lembra uma árvore invertida, assim:



Assinalei novamente as proposições já decompostas. Observe ainda a utilização dos círculos ao redor dos resultados. Quando chegar a uma proposição de uma única constante ou sua negação, os círculos facilitarão essa visualização. Nesse caso, circulei *~P, Q, ~Q* e *R.*

Depois de decompor todas as proposições desse modo, a árvore lógica estará concluída. Cada ramo lhe diz algo sobre uma ou mais interpretações que podem tornar a proposição original verdadeira. Para encontrar essas interpretações, siga o caminho desde o início do tronco até o final do ramo e observe todas as proposições circuladas que encontrar pelo caminho.

Por exemplo, caminhando do início do tronco até o final do primeiro ramo, a única proposição circulada que você encontra é *~P*. Então, esse ramo diz que *qualquer* interpretação em que *P* é falsa fará com que a proposição original seja verdadeira.

Solucionando problemas com as árvores lógicas

Você viu que podemos usar as árvores lógicas para solucionar qualquer problema que possa ser resolvido com as tabelas verdade ou rápidas. Assim como acontece com essas outras ferramentas, as árvores lógicas seguem um processo passo a passo. Veja as etapas da árvore lógica:

1. **Monte.** Para montar uma árvore lógica, construa seu *tronco* de acordo com o tipo de problema que está sendo resolvido.

   O tronco consiste da(s) proposição(ões) que você precisa decompor.

2. **Preencha.** Para preencher a árvore lógica, use as regras de decomposição listadas na Figura 8-1 para construir todos os seus *ramos*.

3. **Interprete.** Para interpretar uma árvore lógica já completa, verifique qual dos dois resultados ocorreu:

   - **Ao menos um ramo está aberto:** Pelo menos uma interpretação faz com que todas as proposições no tronco da árvore lógica sejam verdadeiras.
   - **Todos os ramos estão fechados:** Nenhuma interpretação faz com que todas as proposições do tronco da árvore lógica

sejam verdadeiras (nos tópicos seguintes mostro como isolar um ramo).

Demonstrando Consistência ou Inconsistência

Você pode usar árvores lógicas para descobrir se um conjunto de proposições é consistente ou inconsistente (veja o Capítulo 6 para saber mais sobre consistência). Por exemplo, suponha que você queira descobrir se as três proposições $P \wedge \sim Q$, $Q \vee \sim R$ *e* $\sim P \rightarrow R$ são consistentes ou inconsistentes.

Para decidir se um conjunto de proposições é consistente (ao menos uma interpretação torna todas elas verdadeiras) ou inconsistente (nenhuma interpretação torna todas verdadeiras), construa uma árvore lógica usando o conjunto de proposições como tronco. Veja:

$$P \wedge \sim Q$$

$$Q \vee \sim R$$

$$\sim P \rightarrow R$$

Depois de criar o tronco, você começa a decompor a primeira proposição, $P \wedge \sim Q$. Veja como fica:

$P \wedge \sim Q$ ✓

$Q \vee \sim R$

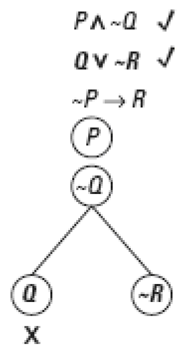
$\sim P \rightarrow R$

P

$\sim Q$

Verifiquei a proposição $P \wedge \sim Q$, depois de tê-la dividido em duas subproposições, e circulei as constantes – P e $\sim Q$.

A proposição seguinte é $Q \vee \sim R$, que dividirei em dois ramos separados, da seguinte forma:



LEMBRE-SE

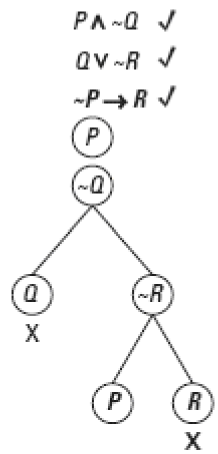
Quando seguir o caminho desde o início do tronco até o fim de um ramo, você passará por um par de proposições contraditórias circuladas, isole o ramo com um X.

Nesse caso, seguindo o caminho desde o início do tronco até o final do ramo da esquerda, você passou por três proposições circuladas *P*, ~*Q* e *Q.* Porém, ~*Q* e *Q* são proposições contraditórias, de forma que tive que isolar esse ramo.

PAPO DE ESPECIALISTA

O fato de isolar esses ramos faz sentido quando você pensa a respeito. Esse ramo lhe diz que qualquer interpretação em que as proposições *P,* ~*Q* e *Q* sejam verdadeiras faz com que todas as proposições originais se tornem verdadeiras. Mas ~*Q* e *Q* não podem ser ambas verdadeiras, de modo que nesse ramo *não* há interpretação possível.

A última proposição a ser decomposta é ~*P* → *R*. Como você já viu na Figura 8-1, os ramos dessa proposição ficarão assim:



Aqui, novamente você pode ver que o ramo que chegou a uma contradição foi fechado. Nesse caso, a contradição é *R e ~R*.

Sua árvore lógica estará terminada quando ocorrer uma das seguintes hipóteses:

> ✓ Todas as proposições ou constantes tiverem sido assinaladas ou circuladas.
>
> ✓ Todos os ramos tiverem sido isolados.

Nesse exemplo, todos os itens foram assinalados (✓) ou circulados, de forma que a árvore está concluída. Estando a árvore terminada, verifique se algum ramo ainda está aberto. A presença ou ausência de ramos abertos em uma árvore lógica já concluída lhe permite determinar se o conjunto de proposições originais é consistente ou inconsistente. Siga estas diretrizes:

> ✓ Se a árvore lógica concluída tiver pelo menos um ramo aberto, o conjunto de proposições é *consistente.*
>
> ✓ Se a árvore lógica concluída tiver todos os ramos fechados, o conjunto de proposições é *inconsistente.*

Como você pode ver, o exemplo neste tópico ainda tem um dos seus ramos abertos, o que significa que existe uma interpretação em que as três proposições são verdadeiras; nesse caso o conjunto é consistente.

Se você quiser saber qual é essa interpretação, basta seguir o trajeto do tronco até o fim do ramo. Quando percorre a extensão da árvore, descobre que os itens circulados são *P,* ~*Q* e ~*R* e a última *P.* Então, a única interpretação que torna todas as três proposições originais verdadeiras é aquela em que o valor de *P* é **V** e os de *Q* e *R* são **F**.

Testando a Validade ou a Invalidade

As árvores lógicas também são úteis quando você quer determinar a validade ou invalidade de um argumento (veja o Capítulo 6 para saber mais sobre a validade). Por exemplo, para descobrir se o seguinte argumento é válido ou inválido:

Premissas:

$\sim P \leftrightarrow Q$

$\sim(P \vee R)$

Conclusão:

$\sim Q \wedge \sim R$

Para decidir se um argumento é válido ou inválido, construa uma árvore lógica usando as premissas e a *negação* da conclusão como tronco.

Usando o exemplo apresentado no início deste tópico, crie um tronco da seguinte forma:

$$\boldsymbol{\sim P \leftrightarrow Q}$$

$$\boldsymbol{\sim(P \vee R)}$$

$$\boldsymbol{\sim(\sim Q \wedge \sim R)}$$

Você não tem que começar sempre pelo topo da árvore. Decompor as proposições em uma ordem diferente muitas vezes é útil. A Figura 8-1 divide as oito formas básicas de proposições em três colunas. Essa divisão ajuda a decidir em que ordem decompô-las. Sempre que tiver mais de uma proposição no tronco de sua árvore lógica, faça a decomposição na seguinte ordem:

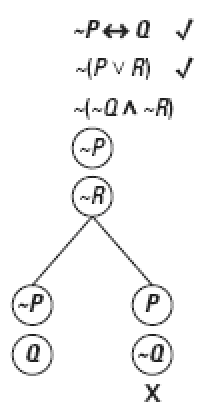
1. Ramo único
2. Ramo duplo com duas subproposições
3. Ramo duplo com uma subproposição

Essa ordem de decomposição faz sentido. Sempre que puder, escolha o caminho do ramo único para que sua árvore fique o menor possível. Porém, quando só houver a opção de ramo duplo, escolha aquele com duas subproposições. Acrescentar duas proposições aumenta as chances de poder isolar um dos ramos.

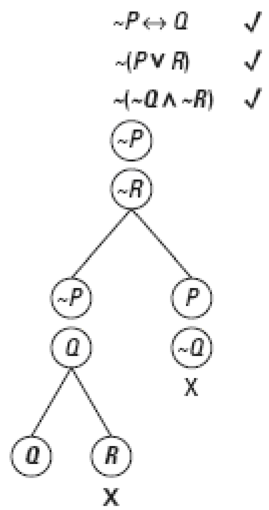
Neste exemplo, somente a segunda proposição, ~(*P* ∨ *R*), o leva a um ramo único. Então, comece por ela, da seguinte forma:

$$\begin{array}{l} {\sim}P \leftrightarrow Q \\ {\sim}(P \vee R) \ \checkmark \\ {\sim}({\sim}Q \wedge {\sim}R) \\ {\sim}P \\ {\sim}R \end{array}$$

As duas proposições restantes levam a ramos duplos. Mas somente a primeira, (~*P* ↔ *Q*), é dividida em duas subproposições. Então ela será a próxima:



Neste ponto, um ramo deve ser isolado: seguindo um caminho do começo do tronco até o final do ramo, você passou por *P e ~P*, que é uma contradição. O último passo, a decomposição de ~(~*Q* ∧ ~*R*), deixa sua árvore assim:



Observe que você terá que continuar a decomposição apenas no ramo aberto, mas não no fechado. Agora todas as proposições estão assinaladas (✓) ou circuladas, o que significa que a árvore está concluída.

LEMBRE-SE

Quando analisar uma árvore lógica para verificar validade ou invalidade, você deve aplicar as seguintes diretrizes:

- ✓ Se a árvore lógica tiver ao menos um ramo aberto, o argumento é *inválido.*
- ✓ Se a árvore lógica tiver todos os ramos fechados, o argumento é *válido.*

No exemplo utilizado neste tópico, um ramo ainda está aberto, o que significa que o argumento é inválido. Seguindo um caminho do começo do tronco até o final desse ramo, você encontra as proposições ~*P, Q* e ~*R*, que estão circuladas. Isso lhe diz que a *única* interpretação em que esse argumento é inválido é aquela em que os valores de *P* e *R* são ambos **F** e o valor de *Q* é **V**.

Classificando Tautologias, Contradições e Contingências

No Capítulo 6, mostrei que toda a proposição na LS é classificada como *tautologia* (que é sempre verdadeira), *contradição* (que é sempre falsa) ou *contingência* (que pode ser verdadeira ou falsa, dependendo do valor de suas constantes). Você pode usar as árvores lógicas para classificar as proposições da LS nessas três categorias.

Tautologias

Suponha que você queira testar se a proposição $((P \wedge Q) \vee R) \rightarrow ((P \leftrightarrow Q) \vee (R \vee (P \wedge \sim Q))$ é uma tautologia.

Para demonstrar que uma proposição é uma tautologia, construa uma árvore lógica usando sua *negação* como tronco.

Usando a negação da proposição anterior, você pode criar um tronco da seguinte forma:

$$\sim(((P \wedge Q)\ 0\ R) \rightarrow ((P \leftrightarrow Q) \vee (R \vee (P \wedge \sim Q)))$$

Ainda que essa proposição pareça grande e complicada, você sabe que ela é uma das oito formas básicas descritas no Capítulo 5 (listadas na Tabela 5-1). Assim, você apenas precisa descobrir qual é a forma correta. Já falei sobre isso no Capítulo 5, mas uma pequena revisão não fará mal nenhum.

O operador principal é o primeiro ~, o único operador que está fora dos parênteses, então a proposição é uma das quatro formas negativas. O operador → abrange o resto da proposição, assim, ela está na forma ~(*x* → *y*). Como você pode ver na Figura 8-1, essa forma é de ramo único, o que levará sua árvore lógica a ficar assim:

$\sim(((P \wedge Q) \vee R) \rightarrow ((P \leftrightarrow Q) \vee (R \vee (P \wedge \sim Q)))$ ✓
$(P \wedge Q) \vee R$
$\sim((P \leftrightarrow Q) \vee (R \vee (P \wedge \sim Q)))$

Observe que removi os parênteses externos da subproposição ((*P* ∧ *Q*) ∨ *R*). Esse é um passo possível, como explicarei no Capítulo 14.

Agora a última proposição está na forma ~(*x* ∨ *y*), que também é de ramo único. Portanto, de acordo com a ordem a ser seguida na decomposição de proposições múltiplas (veja o tópico “Testando a Validade ou a Invalidade”, anteriormente neste capítulo), essa deve ser a próxima proposição a ser trabalhada. Preencha a árvore lógica da seguinte forma:

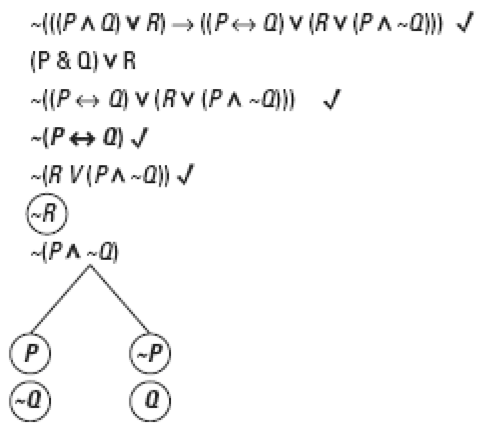
$\sim(((P \wedge Q) \vee R) \rightarrow ((P \leftrightarrow Q) \vee (R \vee (P \wedge \sim Q)))$ ✓
$(P \wedge Q) \vee R$
$\sim((P \leftrightarrow Q) \vee (R \vee (P \wedge \sim Q)))$ ✓
$\sim(P \leftrightarrow Q)$
$\sim(R \vee (P \wedge \sim Q))$

Novamente, a última proposição está na forma ~(*x* ∨ *y*), o que significa que ela é de ramo único e, portanto, o passo seguinte é:

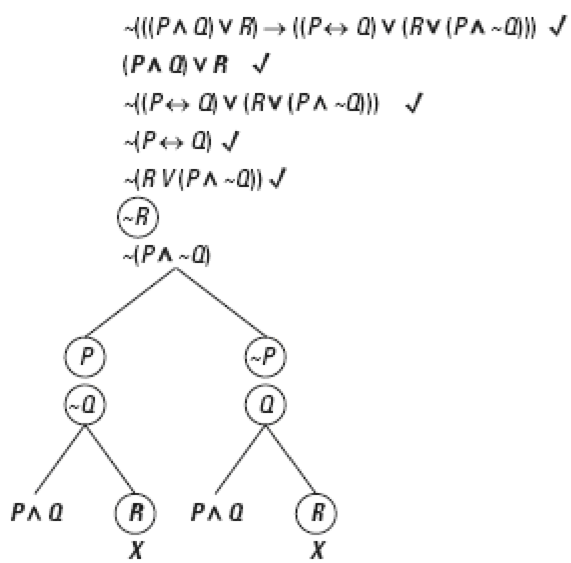
$\sim(((P \wedge Q) \vee R) \rightarrow ((P \leftrightarrow Q) \vee (R \vee (P \wedge \sim Q)))$ ✓
$(P \wedge Q) \vee R$
$\sim((P \leftrightarrow Q) \vee (R \vee (P \,\&\, \sim Q)))$ ✓
$\sim(P \leftrightarrow Q)$
$\sim(R \vee (P \wedge \sim Q))$ ✓
$\sim R$
$\sim(P \wedge \sim Q)$

Embora esse exemplo pareça longo, dê uma olhada e perceba que você já deu três passos sem ramos duplos. Quando eles não aparecem, você economiza grande parte do trabalho, pois somente terá que se preocupar com um ramo e não dois (ou quatro, ou oito!).

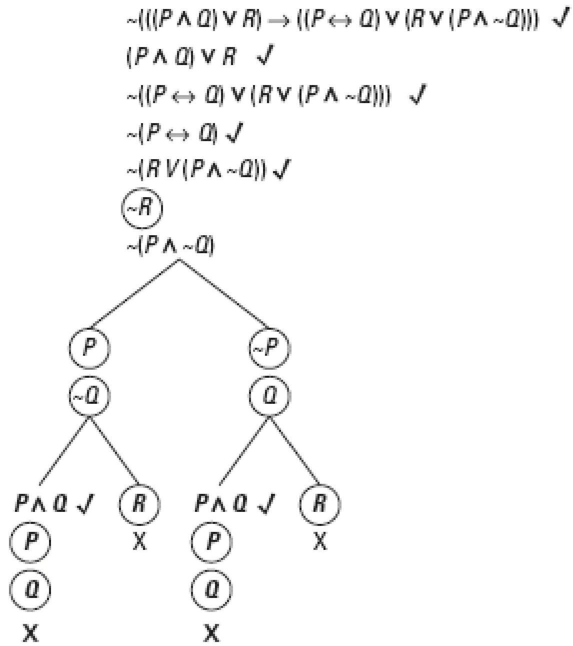
Porém, neste exemplo, você terá, em algum momento, que separar a proposição em um ramo duplo. Tendo em vista que ela se ramifica em duas subproposições, comece com ~(P ↔ Q), de acordo com a ordem de decomposição apresentada no tópico “Testando a Validade e a Invalidade”. A árvore lógica agora ficará assim:



Agora decomponha (P ∧ Q) ∨ R:



Esse passo isola dois dos quatro ramos. Agora P ∧ Q é de ramo único, então decomponha essa proposição nos dois ramos restantes, assim:



Agora você pode fechar todos os ramos remanescentes. Observe que a proposição ~(P ∧ ~Q) *não* está assinalada (✓). Isso não importa, pois após todos os ramos terem sido fechados a árvore estará terminada.

LEMBRE-SE

Para determinar se uma proposição é uma tautologia, siga estas diretrizes:

- ✓ Se a árvore lógica tiver ao menos um ramo aberto, a proposição *não* é tautologia, é uma contradição ou uma contingência (para confirmar ou descartar a contradição como uma das possibilidades, você precisará de outra árvore, como descrito no próximo tópico).
- ✓ Se a árvore lógica não tiver ramos abertos, a proposição é tautologia.

No exemplo deste tópico, a árvore mostra que a proposição é uma tautologia.

Contradições

Suponha que você queira testar a proposição (*P* ↔ *Q*) ^ (~*P* ^ *R*) ^ (*Q* ↔ *R*)) para determinar se é uma contradição.

LEMBRE-SE

Para demonstrar que uma proposição é uma contradição, construa uma árvore lógica usando-a como tronco, que, neste caso, ficará assim:

$$(P \leftrightarrow Q) \wedge (\sim(P \wedge R) \wedge (Q \leftrightarrow R))$$

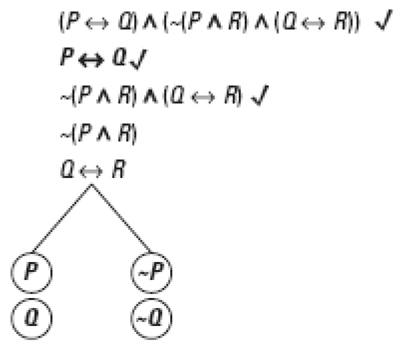
Felizmente, a primeira decomposição desta proposição é de ramo único:

$$\begin{array}{l} (P \leftrightarrow Q) \wedge (\sim(P \wedge R) \wedge (Q \leftrightarrow R)) \;\checkmark \\ P \leftrightarrow Q \\ \sim(P \wedge R) \wedge (Q \leftrightarrow R) \end{array}$$

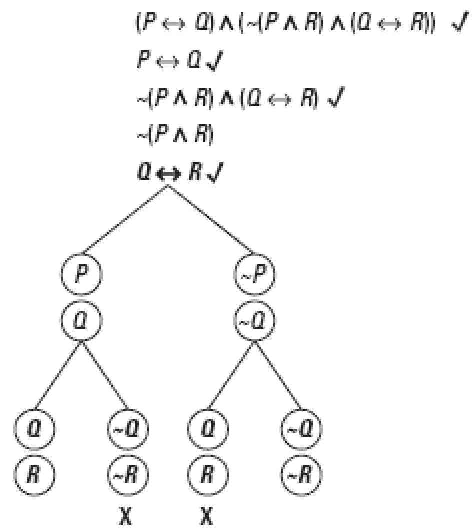
Agora você tem que escolher entre decompor *P* ↔ *Q* ou ~(*P* ^ *R*) ^ (*Q* ↔ *R*). Mas, de acordo com a Figura 8-1, a última proposição é de ramo único, então pegue este caminho primeiro:

$$\begin{array}{l} (P \leftrightarrow Q) \wedge (\sim(P \wedge R) \wedge (Q \leftrightarrow R)) \;\checkmark \\ P \leftrightarrow Q \\ \sim(P \wedge R) \wedge (Q \leftrightarrow R) \;\checkmark \\ \sim(P \wedge R) \\ Q \leftrightarrow R \end{array}$$

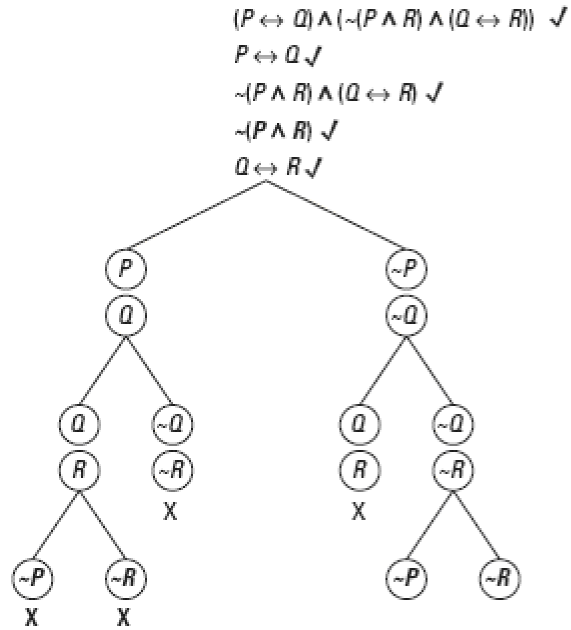
Tendo em vista que não há mais proposições de ramo único, é hora dos ramos duplos, começando com as proposições que geram duas subproposições. Comece com *P* ↔ *Q*:



Agora decomponha *Q* ↔ *R:*



Esse passo fecha dois dos quatro ramos da árvore. Seu último passo então é decompor ~(*P* ∧ *R*) assim:



Agora que todos os ramos estão assinalados (✓) ou circulados, a árvore está concluída.

LEMBRE-SE

Para determinar se uma proposição é uma contradição, aplique estas diretrizes:

- ✓ Se a árvore lógica tiver pelo menos um dos ramos abertos, a proposição *não* é uma contradição, é uma tautologia ou uma contingência (para confirmar ou descartar a tautologia como uma das possibilidades, você precisará de outra árvore, como já falado no tópico anterior).
- ✓ Se a árvore lógica não tiver ramos abertos, a proposição é uma contradição.

Nesse exemplo, dois ramos permaneceram abertos, de forma que a proposição não é uma contradição.

Mesmo com dois ramos abertos na árvore, somente uma interpretação torna a proposição original verdadeira. Siga o caminho do tronco até o final de cada um dos ramos abertos para ver que a interpretação é que *P, Q* e *R* são todas falsas.

Contingências

Verificar se uma proposição é uma contingência, como sempre, é apenas uma questão de excluir as possibilidades de ela ser uma tautologia ou uma contradição (veja o Capítulo 6 para saber mais detalhes sobre a contingência).

Quando você testa uma proposição para verificar se é uma contingência, tem que testá-la como tautologia e como contradição. Se ela *não for* uma tautologia e *não for* uma contradição, tem que ser uma contingência.

Verificando a Equivalência Semântica e a Não Equivalência

Se você tem que analisar um par de proposições para determinar sua equivalência ou não equivalência semântica, está com sorte, pois as árvores lógicas também podem ajudá-lo (como explicado no Capítulo 6, quando duas proposições são semanticamente equivalentes, elas têm o mesmo valor verdade sob todas as interpretações).

LEMBRE-SE

Para decidir se duas proposições são semanticamente equivalentes ou não, você tem que construir duas árvores lógicas:

- ✓ Uma árvore lógica usando a primeira proposição e a negação da segunda como tronco
- ✓ A outra árvore usando a negação da primeira proposição e a segunda como tronco

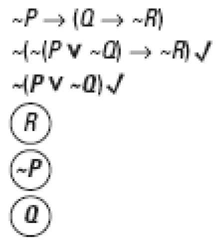
Se você quiser descobrir se as proposições $\sim P \to (Q \to \sim R)$ e $\sim(P \vee \sim Q) \to \sim R$ são semanticamente equivalentes ou não, precisa construir duas árvores, com troncos assim:

Árvore 1:	Árvore 2:
$\sim P \to (Q \to \sim R)$	$\sim(\sim P \to (Q \to \sim R))$
$\sim(\sim(P \vee \sim Q) \to \sim R)$	$\sim(P \vee \sim Q) \to \sim R$

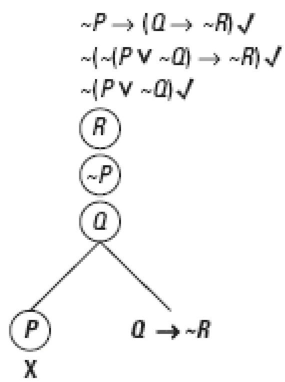
Começando com a Árvore 1, a segunda proposição está na forma $\sim(x \to y)$ de ramo único, então ela será a primeira a ser decomposta:

$$\sim P \to (Q \to \sim R)$$
$$\sim(\sim(P \vee \sim Q) \to \sim R) \checkmark$$
$$\sim(P \vee \sim Q)$$
$$R$$

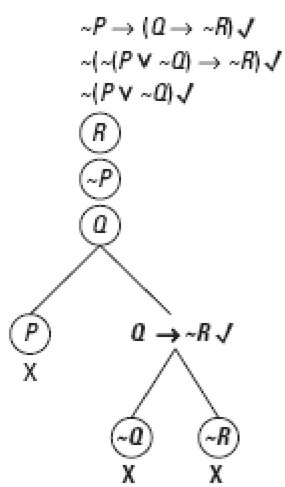
A proposição ~(P ∨ ~Q) também é de ramo único, então trabalhe com ela em seguida:



Depois passe para a primeira proposição, já que a deixou de lado no início:



Este passo isola um dos ramos. Agora decompondo $Q \rightarrow \sim R$, você terá:



A Árvore 1 está terminada.

LEMBRE-SE

Quando você testa duas proposições para determinar se são semanticamente equivalentes ou não, deve seguir as seguintes diretrizes:

- ✓ Se *qualquer* das árvores lógicas tiver um ramo aberto, as proposições são *semanticamente não equivalentes.*
- ✓ Se *ambas* árvores lógicas tiverem todos os ramos fechados, as proposições são *semanticamente equivalentes.*

DICA

Se a primeira árvore tiver pelo menos um ramo aberto, as proposições são *semanticamente não equivalentes*, o que significa que você não precisa passar para a segunda árvore.

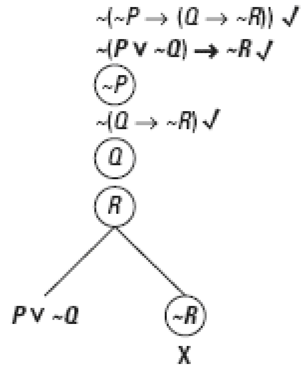
Já que a primeira árvore do exemplo tem todos os ramos fechados, você precisa passar para a próxima árvore. Nesse caso, a primeira proposição é de ramo único, então comece por ela:

$$
\begin{array}{l}
\sim(\sim P \to (Q \to \sim R)) \ \checkmark \\
\sim(P \vee \sim Q) \to \sim R \\
\sim P \\
\sim(Q \to \sim R)
\end{array}
$$

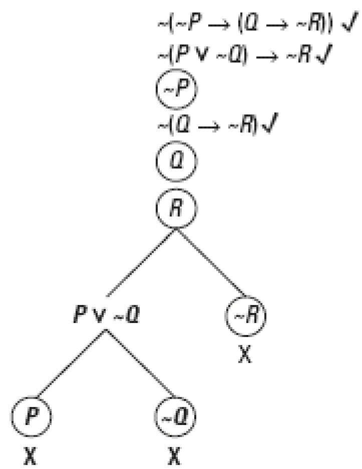
A proposição ~(*Q* → ~*R*) também é de ramo único, então passe para ela:

$$
\begin{array}{l}
\sim(\sim P \to (Q \to \sim R)) \ \checkmark \\
\sim(P \vee \sim Q) \to \sim R \\
\sim P \\
\sim(Q \to \sim R) \ \checkmark \\
Q \\
R
\end{array}
$$

A proposição restante, ~(*P* ˅ ~*Q*) → ~*R*, é de ramo duplo, de forma que o próximo passo ficará assim:



Este passo fecha um ramo. Por fim, decompondo *P* ∨ ~*Q*, você terá:



Todos os ramos das árvores estão fechados, o que significa que as proposições são semanticamente equivalentes.

Parte III:
Provas, Sintaxe e Semântica na LS

“Ele montou umas tabelas verdade, elaborou uns teoremas e, antes que me desse conta, estava comprando um ferro à prova de ferrugem.”

Nesta parte...

Provas são o verdadeiro coração da Lógica. Para alguns estudantes, porém, é aqui que *perdem* a coragem. Mas não tema! As provas não são tão difíceis quanto o fizeram acreditar — basta que você faça a abordagem correta.

Nesta parte, prepare-se para dominar as provas na LS. O Capítulo 9 mostra como são e a forma pela qual são feitas. Você também descobrirá as primeiras oito regras de inferência que compõem o conjunto das regras de implicação. O Capítulo 10 mostra as dez regras de inferência restantes, consideradas o conjunto de regras de equivalência. Os Capítulos 9 e 10 tratam dos métodos de prova direta, enquanto o Capítulo 11 apresenta dois novos métodos de prova: a condicional e a indireta. No Capítulo 12, mostro como e quando usar todas essas ferramentas estudando as estratégias de prova.

Você ainda terá uma visão global da LS. O Capítulo 13 mostrará por que os cinco operadores da LS são suficientes para produzir qualquer função lógica na LS. No Capítulo 14 discutirei uma variedade de tópicos relacionados à sintaxe e à semântica da LS. Aqui também ensinarei como decidir se uma cadeia de símbolos na LS é uma fórmula bem constituída. Finalmente, lhe darei uma pequena amostra da Álgebra Booleana.

Capítulo 9

O que Você Tem que Provar?

Neste Capítulo

- Apresentando a prova direta formal
- Construindo provas usando as regras de inferência

Você já deve ter alguma experiência em fazer provas, aqueles problemas traiçoeiros que eram uma parte importante da Geometria do ensino médio. Nas provas geométricas, você começava com um conjunto de axiomas simples (também chamados de postulados), tais como “todos os ângulos retos são iguais”, e descobria como desenvolvê-los em proposições muito mais complexas, chamadas de teoremas.

A programação de computadores, onde você usa proposições simples para criar softwares complexos, também se parece com o método da prova. Essa ideia de complexidade surgindo da simplicidade é igualmente comum nas provas da Lógica Sentencial (LS).

De certo modo, montar uma prova é como construir uma ponte para atravessar um rio. O ponto de partida é o conjunto de premissas que lhe foram dadas, e o ponto de chegada é a conclusão que você está tentando alcançar. As peças utilizadas para construir a ponte são as *regras de inferência*, um conjunto de 18 maneiras de transformar proposições em outras.

Neste capítulo, apresentarei as primeiras oito regras de inferência, o conjunto de *regras de implicação.* No caminho você descobrirá um pouquinho sobre as provas.

Embora essas regras sejam claras e sem ambiguidades, sua utilização em um caso específico nem sempre é óbvia. Há uma certa arte para a construção de provas, e isso torna o processo interessante e satisfatório quando você consegue acertar, mas é frustrante quando falha. A boa notícia é que existe uma porção de truques, e, uma vez que você os conheça, terá muita munição para usar quando as coisas ficarem complicadas.

Cruzando o Rio Premissa-Conclusão

Um argumento válido é como uma ponte, ele o leva de um ponto (as premissas) a outro (a conclusão), mesmo que águas turbulentas estejam no caminho. As provas lhe mostram um modo de construir essa ponte para que você possa atravessar o rio com segurança.

Por exemplo, considere o seguinte argumento:

Premissas:

$P \to Q$

P

Conclusão:

Q

Já que as provas se concentram tão intensamente nos argumentos, apresento, neste capítulo, como economizar espaço ao escrevê-los. Usando essa convenção você pode escrever o argumento anterior da seguinte forma:

$P \to Q, P : Q$

Como você pode ver, uma vírgula separa as premissas $P \to Q$ e P, e dois pontos são usados antes da conclusão Q. Quando você diz que um argumento é válido, está querendo dizer, na verdade, que a ponte é segura; isto é, se começar pelo lado esquerdo (com as premissas sendo verdadeiras), poderá atravessar a ponte com segurança para o lado direito (a conclusão também será verdadeira).

Veja este breve exemplo da Aritmética, um simples problema de adição:

$2 + 3 = 5$

Essa equação está correta, e se você adicionar 1 de cada lado, ainda terá uma equação correta:

$2 + 3 + \mathbf{1} = 5 + \mathbf{1}$

Verificar se a equação ainda está correta é fácil, porque o resultado da adição nos dois lados é 6.

Agora tente fazer o mesmo usando variáveis:

$a = b$

Se adicionar uma terceira variável aos dois lados da equação:

$a + \mathbf{c} = b + \mathbf{c}$

A equação estará correta? Você não tem como afirmar isso com certeza, pois não sabe o significado das variáveis. Porém, mesmo sem saber seus significados, você pode dizer que se a primeira equação estiver correta, a segunda também *deverá* estar.

Em consequência, a seguinte proposição é válida:

$a = b \rightarrow a + c = b + c$

Com essa informação você pode construir o seguinte argumento:

Premissas:

$a = b$

$a = b \rightarrow a + c = b + c$

Conclusão:

$a + c = b + c$

LEMBRE-SE

Sabendo que o lado esquerdo de um argumento válido é verdadeiro, você está seguro para atravessar para o lado direito, uma vez que ele também é verdadeiro. Não importa os números que usar, se você partir de uma proposição verdadeira e conservar a *forma* geral do argumento, sempre acabará com uma conclusão verdadeira.

Usando as Oito Regras de Implicação na LS

Provas da LS funcionam tão bem quanto as da Aritmética. A única diferença é que, ao invés de usar símbolos aritméticos, usa-se os operadores da LS, que você já conhece e aprendeu a amar (veja o Capítulo 4 para saber mais sobre esses adoráveis e divertidos operadores).

A LS possui oito *regras de implicação* que lhe permitem construir a ponte para atravessar de um lado para outro. Em outras palavras, você começa com uma ou mais proposições verdadeiras e termina com outra. A maioria dessas regras é simples e pode até parecer banal. Porém, essa simplicidade é sua força, pois é o que o permite lidar com ideias muito mais complexas com muita segurança.

Neste tópico, apresento as oito regras de implicação e mostro como utilizálas para construir provas. Nesse trajeto, você aprenderá alguns truques para facilitar a tarefa (veja o Capítulo 12 para conhecer mais profundamente as estratégias de prova). Ainda verá várias maneiras de memorizar essas regras para usar sempre que precisar.

No fim das contas, você terá que decorar essas regras. Mas, por ora, trabalhe com elas e acabará descobrindo, ao final, que já gravou a maioria delas sem muito esforço.

As regras → : Modus Ponens e Modus Tollens

No Capítulo 3 comparei proposições *se* a um escorregador: se a primeira parte da proposição é verdadeira, a segunda também *deve* ser, para que ela seja verdadeira como um todo. As duas regras de implicação deste tópico — *Modus Ponens* (**MP**) e *Modus Tollens* (**MT**) — aproveitam essa ideia de diferentes maneiras.

Modus Ponens (MP)

A **MP** usa a ideia do escorregador de maneira direta: ela afirma que: “se você sabe que a proposição da forma $x \rightarrow$ y é verdadeira e que a parte *x também o* é, isso o levará à conclusão de que a parte y é igualmente verdadeira”. Em outras palavras, uma vez que você sobe no escorregador em *x*, não tem outra escolha senão chegar até *y.*

MP: $x \rightarrow y, x : y$

A **MP** diz que *qualquer* proposição desse formato é válida. Esta é sua forma simplificada:

$P \rightarrow Q, P : Q$

Mas pela mesma regra, esses argumentos semelhantes também são todos válidos:

$\sim P \rightarrow \sim Q, \sim P: \sim Q$

$(P \wedge Q) \rightarrow (R \wedge S), P \wedge Q : R \wedge S$

$(P \leftrightarrow \sim(Q \wedge \sim R)) \rightarrow (\sim S \vee (R \rightarrow P)), (P \leftrightarrow \sim(Q \wedge \sim R)) : (\sim S \vee (R \rightarrow P))$

Uma prova típica tem diversas linhas numeradas. As de cima são as premissas, a última é a conclusão e as linhas entre elas são os passos intermediários que as conectam logicamente. Cada uma tem um número e uma proposição, seguida pela sua justificativa (o que inclui a regra e as linhas que ela atinge).

Veja, por exemplo, a prova para o argumento $(P \wedge Q) \rightarrow R, (P \wedge Q) : R$:

1. $(P \wedge Q) \rightarrow R$	**P**
2. $(P \wedge Q)$	**P**
3. R	1, 2 **MP**

Como você pode ver, essa prova não requer qualquer passo intermediário. As premissas (**P**) levam imediatamente à conclusão, com a justificativa da regra **MP** aplicada nas linhas 1 e 2.

Agora tente provar que esse argumento, um pouco mais complicado, é válido:

$P \rightarrow Q, Q \rightarrow R, R \rightarrow S, P : S$

O primeiro passo para qualquer prova é sempre o mesmo: copie todas as premissas com os números das linhas e as justificativas. Veja como deve ficar:

1. $P \rightarrow Q$	**P**
2. $Q \rightarrow R$	**P**
3. $R \rightarrow S$	**P**
4. P	**P**

Depois de copiar as premissas, observe o próximo passo possível. Nesse caso pode usar **MP** com as linhas 1 e 4:

5. Q	1, 4 **MP**

A **MP** permite a criação de uma nova proposição, *Q*, que pode ser usada no próximo passo. Desta vez pode usar a **MP** com as linhas 2 e 5:

6. R	2, 5 **MP**

Novamente, pode criar uma nova proposição, *R*, que será usada no próximo passo. O último passo praticamente já está pronto:

7. S	3, 6 **MP**

Você sabe que acabou a prova quando a conclusão do argumento aparece. Nesse caso, *S* é a conclusão que está tentando justificar, de modo que a prova está terminada.

Modus Tollens (MT)

A **MT** usa a ideia do escorregador de modo diferente da **MP**: ela afirma que “se você sabe que uma proposição no formato $x \rightarrow y$ é verdadeira e que y é falsa, pode concluir que x também é falsa”. Em outras palavras, se *não* chegou ao final do escorregador em y, então *não* subiu nele em x.

MT: $x \rightarrow y, \sim y : \sim x$

Assim como na **MP** (mostrada no tópico anterior), a **MT** e todas as outras regras de inferência podem ser generalizadas. Portanto, alguns argumentos válidos facilmente prováveis nesse formato são:

$P \rightarrow Q, \sim Q : \sim P$

$(P \wedge Q) \rightarrow R, \sim R : \sim(P \wedge Q)$

$(P \vee Q) \rightarrow (R \leftrightarrow S), \sim(R \leftrightarrow S) : \sim(P \vee Q)$

Conhecendo essa regra, agora você pode provar a validade desse argumento:

$P \rightarrow Q, \sim P \rightarrow R, \sim Q : R$

Como sempre, comece com a cópia das premissas:

1. $P \rightarrow Q$	**P**
2. $\sim P \rightarrow R$	**P**
3. $\sim Q$	**P**

Até agora você conhece duas regras, a **MP** e a **MT**,e precisará de ambas nessa prova.

Na maioria das provas, é mais fácil trabalhar com as proposições curtas do que com as longas.

A proposição mais curta aqui é ~*Q*, de modo que um bom plano é procurar uma forma de usá-la. Para não ter que procurar muito, use a **MT** da seguinte forma:

4. ~*P* 1, 3 **MT**

Agora use a **MP**:

5. *R* 2, 4 **MP**

A prova está concluída quando a conclusão aparece.

As regras ∧ : Conjunção e Simplificação

As duas regras ∧ estão relacionadas por um fator em comum: as proposições ∧. A *simplificação* (**Simp**) é útil para decompô-las no início da prova, e a *Conjunção* (**Conj**) para construí-las, ao final.

Conjunção (Conj)

A **Conj** diz que: “se você conhece duas coisas separadamente, conhece ambas também juntas”.

Conj: *x, y* : *x* ∧ *y*

A Conj é uma forma muito direta: se você tem duas proposições, x e y, pode concluir que a proposição $x \wedge y$ também é verdadeira.

Experimente essa prova em proposições maiores:

$$P \rightarrow Q,\ R \rightarrow S,\ P,\ {\sim}S : (P \wedge {\sim}S) \wedge (Q \wedge {\sim}R)$$

Primeiro copie as premissas:

1. $P \rightarrow Q$	**P**
2. $R \rightarrow S$	**P**
3. P	**P**
4. ${\sim}S$	**P**

Escrever uma prova pode ser bem parecido com uma caça ao tesouro: tente encontrar o caminho para a próxima pista da forma que puder. Por exemplo, analise a conclusão para saber onde está tentando chegar. Então olhe as premissas e veja quais podem ajudá-lo.

De um modo geral, conclusões longas tendem a resultar em provas mais fáceis do que as curtas. A estratégia é tentar construir uma conclusão longa, uma subproposição de cada vez.

Nessa prova você quer construir as subproposições $(P \wedge {\sim}S)$ e $(Q \wedge {\sim}R)$. Seus operadores são do tipo $\wedge$, o que lhe dá a dica de que uma **Conj** pode ajudá-lo a construí-las. Na verdade, você pode obter uma delas em apenas um passo:

5. $P \wedge {\sim}S$	3, 4 **Conj**

Procure por proposições que tenham constantes em comum e veja se consegue combiná-las usando as regras de inferência.

Analisando o exemplo, você pode ver que as linhas 1 e 3 têm constantes em comum, assim como a 2 e a 4:

6. Q	1, 3 **MP**
7. $\sim R$	2, 4 **MT**

Agora você pode combinar essas proposições usando a **Conj**:

8. $Q \wedge \sim R$	6, 7 **Conj**

Isso lhe dá a outra subproposição da conclusão. A única coisa que falta fazer é construir a conclusão usando as peças que você coletou:

9. $(P \wedge \sim S) \wedge (Q \wedge \sim R)$	5, 8 **Conj**

Essa foi uma prova longa, de nove linhas, mas trabalhando nela, peça por peça, tudo se encaixa.

Agora que você fez a prova passo a passo, copie o argumento, feche o livro e veja se consegue refazê-la sozinho. Pode ser que encontre algumas pedras pelo caminho, mas é melhor descobrir agora do que na hora de uma prova de Lógica!

Simplificação (Simp)

A **Simp** lhe diz que: “se você tem duas coisas juntas, também terá uma delas separadamente”.

Simp: $x \wedge y : x$

$x \wedge y : y$

A **Simp** é um tipo avesso da **Conj**. Mas, ao invés de começar com as peças para construir o todo, você começa pelo todo e o reduz a uma das duas partes.

Tente escrever a prova para este argumento:

$P \to Q, R \to S, P \wedge \sim S : Q \wedge \sim R$

Como sempre, copie as premissas, assim:

1. $P \to Q$	**P**
2. $R \to S$	**P**
3. $P \wedge \sim S$	**P**

Use a **Simp** no começo de uma prova para *desmembrar* proposições $\wedge$, isto é, transforme-as em porções menores. Isso resultará em proposições mais fáceis de trabalhar, que podem ser usadas para construir a conclusão.

A linha 3 é sua única oportunidade de usar essa regra:

4. P	3, **Simp**
5. $\sim S$	3, **Simp**

Você deve ter percebido uma certa semelhança entre as provas apresentadas neste capítulo. Se percebeu, é bom! Os próximos dois passos são muito familiares agora:

6. Q 1, 4 **MP**

7. $\sim R$ 2, 5 **MT**

Agora tudo o que falta é juntar as peças, assim:

8. $Q \wedge \sim R$ 6, 7 **Conj**

As regras ∨: Adição e Silogismo Disjuntivo

Assim como a **Simp** está relacionada à **Conj**, o silogismo disjuntivo (**SD**) se relaciona com a adição (**Add**). Ambas as regras trabalham com proposições do tipo ∨. O **SD** as desconstrói e a **Add** as reconstrói.

Adição (Add)

A **Add** diz que: “se você conhece *x*, pode concluir tanto *x quanto y*”.

Add: $x : x \vee y$

À primeira vista, essa regra pode parecer absolutamente estranha. Você pode estar se perguntando: “se estou começando apenas com o *x*, de onde terá surgido o *y*?” Acredite ou não, a beleza da **Add** é justamente o surgimento aparentemente mágico do *y.*

Lembre-se de que, para construir uma proposição do tipo ∨ verdadeira, você só precisa que uma de suas partes seja verdadeira (veja o Capítulo 4). A outra parte pode ser qualquer coisa.

Experimente esta prova:

$$Q \rightarrow S, Q : ((P \leftrightarrow \sim Q) \leftrightarrow R) \vee ((P \vee S) \wedge (Q \vee R))$$

Essa prova parece mesmo esquisita. Não há muito com o que trabalhar, apenas duas premissas:

1. $Q \to S$ **P**

2. Q **P**

O primeiro passo aparece sozinho:

3. S 1, 2 **MP**

Quando a conclusão de um argumento é uma proposição $\vee$, você apenas precisa construir uma das duas subproposições e, então, usar a **Add** para acrescentar o resto.

A peça-chave para entender aqui é que você tem duas escolhas: pode provar a primeira parte da conclusão — $((P \leftrightarrow \sim Q) \leftrightarrow R)$— ou a segunda —$((P \vee S) \wedge (Q \vee R))$. Aposte na segunda parte, pois você ainda não conhece as regras para lidar com proposições $\leftrightarrow$.

Uma prova é como uma ponte: quanto maior ela é, mais provável é que tenha que ser construída a partir dos dois lados e não apenas de um. Então, para as provas mais difíceis como essas escreva a conclusão a que está tentando chegar no final da página e trabalhe de baixo para cima.

Nesse caso a conclusão ficará assim:

6. $(P \vee S) \wedge (Q \vee R)$

7. $((P \leftrightarrow \sim Q) \leftrightarrow R) \vee ((P \vee S) \wedge (Q \vee R))$ 6 **Add**

Com essa configuração, você está basicamente dizendo: “ainda não tenho certeza de como cheguei aqui, mas o último passo foi juntar aquela proposição $\leftrightarrow$ usando a **Add**”. Aliás, não se preocupe demais com os números das linhas quando estiver trabalhando de baixo para cima – é que eu tenho dons sobrenaturais.

Agora observe a linha 6. De novo, quando se trabalha de baixo para cima, a pergunta a ser feita é: “como cheguei aqui?”. Agora perceba que ($P \vee S$) ∧ ($Q \vee R$) é uma proposição ∧; uma das formas de construí-la é juntando as duas partes com a **Conj:**

4. $P \vee S$	
5. $Q \vee R$	
6. $(P \vee S) \wedge (Q \vee R)$	4, 5 **Conj**
7. $((P \leftrightarrow {\sim}Q) \leftrightarrow R) \vee ((P \vee S) \wedge (Q \vee R))$	6 **Add**

De novo, você não sabe exatamente *como* chegou aqui, mas se descobrir um caminho para construir as proposições $P \vee S$ e $Q \vee R$, o restante aparecerá.

Agora é mais fácil visualizar como a mágica aconteceu afinal, pois construir $P \vee S$ e $Q \vee R$ não é muito difícil. Observe as linhas 2 e 3 e use a **Add** para ligar as lacunas. Veja a prova inteira, do princípio ao fim:

1. $Q \rightarrow S$	**P**
2. Q	**P**
3. S	1, 2 **MP**
4. $P \vee S$	2, **Add**
5. $Q \vee R$	3, **Add**
6. $((P \vee S) \wedge (Q \vee R))$	4, 5 **Conj**
7. $((P \leftrightarrow {\sim}Q) \leftrightarrow R) \vee ((P \vee S) \wedge (Q \vee R))$	6 **Add**

Silogismo Disjuntivo (SD)

O **SD** afirma que: “se você tem duas opções e pode eliminar uma, pode ter certeza daquela que restou”.

SD: $x \vee y, \sim x : y$

$x \vee y, \sim y : x$

O **SD** está relacionado à **Add** do seguinte modo: o **SD** decompõe uma proposição $\vee$ e a **Add** a constrói.

Veja o que você pode fazer com o seguinte argumento:

$P \rightarrow \sim Q, P \vee R, Q \vee S, \sim R : \sim P \vee S$

Primeiro copie as premissas:

1. $P \rightarrow \sim Q$	**P**
2. $P \vee R$	**P**
3. $Q \vee S$	**P**
4. $\sim R$	**P**

Nesse caso, a proposição mais simples é $\sim R$. A linha 2 também contém a constante R. Essas duas proposições lhe permitem usar o **SD** imediatamente:

5. P	2, 4 **SD**

Agora que você tem a proposição P, a próxima surgirá facilmente:

6. $\sim Q$	1, 5 **MP**

Você tem novamente a oportunidade de usar o **SD**:

7. S	3, 6 **SD**

Finalmente, você não pode perder a oportunidade de usar a **Add**:

8. $\sim P \vee S$ 7 **Add**

As Regras → Duplas: Silogismo Hipotético e Dilema Construtivo

O *Silogismo Hipotético* (**SH**) e o *Dilema Construtivo* (**DC**) permitem a elaboração de conclusões quando você começa com duas proposições →. Você não as usará tanto quanto as outras seis regras mencionadas neste capítulo, mas ainda assim precisará delas de vez em quando.

Silogismo Hipotético (SH)

O **SH** faz sentido quando você o olha. Ele lhe diz: “se você sabe que *x* leva a *y* e que *y* leva a *z*, então *x* leva a *z*”.

SH: $x \to y,\ y \to z : x \to z$

Observe que o **SH** é a primeira regra até agora que não contém constantes únicas. Ele não desconstrói nem constrói proposições.

Experimente este exemplo:

$$P \to Q,\ Q \to R,\ R \to S,\ \sim S : \sim P \wedge (P \to S)$$

Como sempre, escreva suas premissas:

1. $P \to Q$ **P**

2. $Q \to R$ **P**

3. $R \to S$ **P**

4. $\sim S$ **P**

Analisando a conclusão, você vê que precisará pegar as duas partes – $\sim P$ e $P \to S$ – e juntá-las usando uma **Conj**. Pode fazer isso em qualquer ordem. Comece usando a **SH**:

5. $P \rightarrow R$	1, 2 **SH**
6. $P \rightarrow S$	3, 5 **SH**

Isso lhe dá a primeira parte. Agora, a segunda não está difícil de encontrar:

7. $\sim P$	4, 6 **MT**

Então, basta juntá-las usando a **Conj**:

8. $\sim P \wedge (P \rightarrow S)$	5, 7 **Conj**

Dilema Construtivo (DC)

O **DC** é menos intuitivo do que as outras regras deste capítulo, até que você pare para pensar nele em português simples, ele diz “se você sabe que tem *w* ou *x*, e também sabe que *w* o leva a *y* e *x* o leva a *z*, então, você tem *y* ou *z*”.

DC: $w \vee x, w \rightarrow y, x \rightarrow z : y \vee z$

Essa regra é também a única que usa três proposições para produzir uma, o que resulta em poucas possibilidades de usar o **DC**. No entanto, quando essa oportunidade surge, geralmente ela está bem na sua frente, piscando como um sinal de neon. Em outras palavras, é superfácil de encontrar.

Nesse exemplo lhe dei seis premissas, numa tentativa desesperada de camuflar essa oportunidade:

$P \to Q, Q \to R, S \to T, U \to V, S \vee U, \sim R : (\sim P \wedge \sim Q) \wedge (T \vee V)$

Então, primeiro escreva as premissas:

1. $P \to Q$	**P**
2. $Q \to R$	**P**
3. $S \to T$	**P**
4. $U \to V$	**P**
5. $S \vee U$	**P**
6. $\sim R$	**P**

À primeira vista, esse exemplo é só uma gigantesca bagunça. Um coisa que deve ser observada, porém, é que a conclusão é uma proposição $\wedge$. Isso significa que se você conseguir extrair suas duas partes – $\sim P \wedge \sim Q$ e $T \vee V$ –, poderá usar a **Conj** para chegar ao todo.

Primeiro use o **DC** para obter $T \vee V$:

7. $T \vee V$	3, 4, 5 **DC**

Agora, como fazer para obter $\sim P \wedge \sim Q$? É óbvio que basta obter $\sim P$ *e* $\sim Q$. Não parece difícil, não é mesmo? Observe os passos seguintes:

8. $\sim Q$	2, 6 **MT**
9. $\sim P$	1, 8 **MT**
10. $\sim P \wedge \sim Q$	8, 9 **Conj**

Para concluir a prova, basta juntá-las usando a **Conj**, assim:

11. $(\sim P \wedge \sim Q) \wedge (T \vee V)$	7, 10 **Conj**

Capítulo 10

Equivalência de Oportunidades: Colocando as Regras de Equivalência em Prática

Neste Capítulo

- Entendendo a diferença entre as regras de equivalência e de implicação
- Usando dez importantes regras de equivalência

Se você gosta das regras de implicação (veja o Capítulo 9), vai adorar as outras dez regras de inferência — as *de equivalência* —, explicadas neste capítulo. Por quê? Vou lhe mostrar os motivos.

Primeiro, essas regras irão impressioná-lo e encantar (e a muitos outros admiradores) pela facilidade com que trabalham com os problemas de Lógica. Por exemplo, tente provar a validade do argumento a seguir usando somente as regras de implicação:

$$\sim(P \wedge Q), P : \sim Q$$

Infelizmente é impossível. Mas, por sorte, com as novas e melhoradas regras de equivalência apresentadas neste capítulo, problemas como esse serão apenas distrações corriqueiras em um dia ensolarado. Ainda há outras boas notícias: regras de equivalência são, em geral, mais fáceis e mais flexíveis de serem usadas nas provas, por inúmeros motivos importantes, que também serão tratados aqui.

Neste capítulo, você descobrirá como aplicar dez importantes regras de equivalência; receberá dicas de quando e como usá-las e continuará a afiar suas habilidades de provar a validade dos argumentos. As provas neste capítulo também usam as regras de implicação.

Distinguindo Implicações e Equivalências

As regras de implicação discutidas no Capítulo 9 têm inúmeras limitações importantes que as de equivalência, tratadas neste capítulo, *não* têm. Por isso elas são consideradas mais flexíveis e, de modo geral, mais úteis do que as de implicação. Continue lendo para descobrir as diferenças entre esses dois conjuntos de regras.

Pensando nas equivalências como via de duas mãos

Uma das mais importantes diferenças entre equivalência e implicação é a forma como elas agem: a equivalência funciona em ambas as direções, enquanto que a implicação somente em uma.

Por exemplo, quando você sabe que $x \rightarrow y$ e x são verdadeiras, a Modus Ponens (**MP**) lhe diz que y também é verdadeira (veja o Capítulo 9). No entanto, a inversão dessa dedução gera problemas, porque saber que y é verdadeira não é certamente informação suficiente para você decidir que $x \rightarrow y$ e x são verdadeiras.

Por sorte essa limitação não se aplica às dez regras de equivalência. Essas regras lhe dizem que duas proposições são *intercambiáveis*: sempre que puder usar uma poderá usar a outra e vice-versa.

Aplicando equivalências às partes do todo

Outra diferença entre as regras de equivalência e as de implicação é que as primeiras podem ser aplicadas às partes de uma proposição, o que não ocorre com as de implicação. Veja um exemplo de um erro claro quando se trabalha com as implicações:

1.	$(P \wedge Q) \rightarrow R$	**P**
2.	P	1, **Simp**

(ERRADO

Lembre-se de que a regra de simplificação (**Simp**) afirma que $x \wedge y : x$. Dessa forma, o erro aqui é pensar que se pode aplicar a regra de implicação **Simp** em uma parte da linha 1 ($P \wedge Q$). As regras de equivalência, porém, não se prendem a essas restrições.

Descobrindo as Dez Equivalências Válidas

Deixe-me adivinhar, você está ansioso para conhecer essas dez regras de equivalência que, aliás, terá que decorar. Aqui estão elas, completas e com exemplos.

DICA

Uma pequena observação que você precisa saber é sobre o símbolo **::** (dois pontos duplos). Quando colocado entre duas proposições, esse símbolo significa que ambas são equivalentes, isto é, você pode substituir uma pela outra, sempre que necessário.

Dupla Negação (DN)

A **DN** é simples, pois declara que "se *x* é verdadeira, então *não x* também é verdadeira".

DN: $x :: {\sim}{\sim}x$

Se você leu o Capítulo 9, já deve saber, de olhos fechados, como trabalhar com a prova a seguir, sem usar qualquer das regras de equivalência:

$\sim P \to Q, \sim Q : P$

1.	$P \to Q$	**P**
2.	$\sim Q$	**P**
3.	P	1, 2 **MT**

No entanto, toda vez que você negar uma negação — por exemplo, quando nega $\sim P$ e a transforma em *P* —, tecnicamente, precisa seguir os seguintes passos:

1. Negar $\sim P$ substituindo-o por $\sim\sim P$.
2. Usar a Dupla Negação (**DN**) para transformar $\sim\sim P$ em *P.*

Assim, a prova no início deste tópico tem, tecnicamente, um passo faltando. Veja a versão seguinte:

$\sim P \to Q, \sim Q : P$

1.	$\sim P \to Q$	**P**
2.	$\sim Q$	**P**
3.	$\sim(\sim P)$	1, 2 **MT**
4.	P	3 **DN**

Veja se seu professor defende essa tecnicalidade. Na hipótese afirmativa, tenha cuidado de não presumir uma **DN** sem explicitá-la em sua prova.

Estou em um caminho delicado aqui: ao longo deste capítulo usarei a **DN** com frequência, sem explicitá-la. Minha filosofia é a de que você não precisa perder as estribeiras toda vez que tiver que trocar o negativo pelo positivo. Então, pode me processar se quiser, mas *não* sei *nada* sobre Dupla Negação.

Contraposição (Contra)

O Capítulo 4 mostra que uma proposição na forma $x \to y$ e sua contraposição ($\sim y \to \sim x$) sempre têm o mesmo valor verdade. O Capítulo 6 indica que duas proposições com o mesmo valor verdade são semanticamente equivalentes. Esses dois fatos juntos lhe dão a **Contra**.

Contra: $x \to y :: \sim y \to \sim x$

A **Contra** está relacionada à Modus Tollens (**MT**), tratada no Capítulo 9. Cada uma dessas regras diz: "quando você parte de um ponto em que sabe que a proposição $x \to y$ é verdadeira, então o fato de $\sim y$ ser verdadeira o leva rapidamente a $\sim x$".

Um modo fácil de pensar na **Contra** é: *inverta* e *negue as* *duas*. Isto é, você pode *inverter* as duas partes de uma proposição $\to$ se *negar ambas as* suas partes. Por exemplo, observe a seguinte prova:

$P \to Q, \sim P \to R : \sim R \to Q$

1.	$P \to Q$	**P**
2.	$\sim P \to R$	**P**

Essa prova lhe dá duas oportunidades de usar a **Contra:**

3.	$\sim Q \to \sim P$	1 **Contra**
4.	$\sim R \to P$	2 **Contra**

Agora você pode concluir a prova em um passo, usando a **SH**:

5.	$\sim R \to Q$	1, 4 **SH**

Aliás, observe que nessa prova você não usa a linha 3 e, na verdade, pode eliminá-la se quiser (a princípio, seu professor não vai lhe tirar pontos por passos desnecessários, mas se você não tiver certeza disso, então elimine-a).

Você não tem que usar todas as proposições, nem mesmo todas as premissas (embora na maioria das vezes precisará de todas). Mas pode ser útil escrever todas as proposições que puder para depois ver se vai precisar delas. Você aprenderá mais sobre essa estratégia de escrever todas as informações fáceis no Capítulo 12.

Implicação (Impl)

O racional por trás da **Impl** é simples: quando você sabe que a proposição $x \to y$ é verdadeira, sabe que ou x é falsa (isto é, $\sim x$ é verdadeira) ou y é verdadeira. Em outras palavras, sabe que a proposição $\sim x \vee y$ é verdadeira.

Impl: $x \to y :: \sim x \vee y$

O que você provavelmente não sabia, porém, é que essa regra também funciona às avessas (assim como todas as equivalências válidas). Assim, sempre que tiver $x \vee y$, poderá mudar para $\sim x \to y$. Os professores mais detalhistas podem exigir que primeiro você troque $x \vee y$ para $\sim\sim x \vee y$, usando a **DN** e depois para $\sim x \to y$, usando a **Impl.**

Um modo fácil de pensar na **Impl** é: *troque e negue a primeira.* Por exemplo, troque uma proposição → para uma ∨ (ou vice-versa), desde que você *negue* a *primeira parte* da proposição. Observe esta prova:

$P \to Q, P \vee R : Q \vee R$

1.	$P \to Q$	**P**
2.	$P \vee R$	**P**

Como você pode ver, essa é uma prova difícil, pois você não tem muitas opções e nem proposições de constantes simples.

A **Impl** liga todas as proposições ∨ com uma do tipo →. A **Contra** lhe dá duas versões de toda as proposições →. Essas são três formas de se trabalhar.

Por exemplo, $P \to Q$ tem duas formas equivalentes: $\sim P \vee Q$ por **Impl** e $\sim Q \to \sim P$ por **Contra**. Escrever as duas não custa nada e nem é obrigatório, e visualizá-las pode lhe sugerir algo, então certifique-se de escrever todos os passos quando os encontrar. Aqui estão eles:

3.	$\sim P \vee Q$	1 **Impl**
4.	$\sim Q \to \sim P$	1 **Contra**

O mesmo é verdade em relação a $P \vee R$:

5.	$\sim P \rightarrow R$	2 **Impl**
6.	$\sim R \rightarrow P$	2 **Contra**

Agora, analise o que você está tentando provar: $Q \vee R$. Com a **Impl**, isso é o mesmo que $\sim Q \rightarrow R$. Aha! Aqui estão os próximos passos:

7.	$\sim Q \rightarrow R$	4, 5 **SH**
8.	$Q \vee R$	7 **Impl**

Você não precisava das linhas 3 e 6, mas são de graça, sendo assim, quem se importa? (Se a resposta para essa pergunta retórica for “Meu professor – ele tirará pontos por passos extras”, então volte e elimine-as.)

DICA

Quando estiver em dúvida em uma prova, escreva todas as proposições que conseguir. Não fará mal algum e, geralmente, ajuda. Chamo essa estratégia de *x-tudo*, porque você monta o problema com tudo a que tem direito. Discuto essa e outras estratégias de prova com mais detalhes no Capítulo 12.

Exportação (Exp)

A **Exp** não é intuitiva até que você a escreva (o que farei por você, neste momento), o que faz com que seu significado simplesmente salte aos olhos.

Exp: $x \rightarrow (y \rightarrow z) :: (x \wedge y) \rightarrow z$

Para entender essa regra, pense em um exemplo onde $x \rightarrow (y \rightarrow z)$ fosse a forma de uma proposição em português. Veja uma possibilidade:

> *Se* eu for trabalhar hoje, *então* se eu vir meu chefe, *então* pedirei um aumento.

Agora pense nesse exemplo parecido com $(x \wedge y) \rightarrow z$:

Se eu for trabalhar hoje *e* vir meu chefe, *então* pedirei um aumento.

Essas duas proposições significam, na essência, a mesma coisa, o que lhe diz que as duas formas de proposição em que são baseadas são semanticamente equivalentes.

Não faça confusão com a posição dos parênteses! A **Exp** não lhe diz nada sobre a proposição $(x \to y) \to z$ ou $x \wedge (y \to z)$.

Experimente esta prova:

$(P \wedge Q) \to R, \sim R \vee S, P : Q \to S$

1.	$(P \wedge Q) \to R$	**P**
2.	$\sim R \vee S$	**P**
3.	P	**P**

A proposição 1 é uma das duas formas em que a **Exp** funciona, experimente:

4.	$P \to (Q \to R)$	1, **Exp**

Agora outro caminho se abre:

5.	$Q \to R$	3, 4 **MP**

Certo, aí você tem $Q \to R$, mas, agora, precisa de $Q \to S$. Basta conseguir chegar à $R \to S$ para poder construir a ponte de que precisa para usar **SH**. Observe esses passos:

6.	$R \to S$	2 **Impl**
7.	$Q \to S$	5, 6 **SH**

Comutação (Comm)

Você deve se lembrar da propriedade comutativa da Aritmética como sendo a regra que lhe diz que a ordem dos fatores não altera o produto, tanto na adição quanto na multiplicação, por exemplo: 2 + 3 = 3 + 2 e 5 x 7 = 7 x 5.

Na LS a **Comm** lhe diz que a ordem não afeta as operações envolvendo ∧ e ∨. Assim, a **Comm** tem duas versões:

Comm: $x \wedge y :: y \wedge x$

$x \vee y :: y \vee x$

Como na **DN**, a **Comm** parece que, de tão óbvia, nem precisa ser mencionada. Mas, ao contrário da **DN**, acredito que é sempre melhor explicitar o uso da **Comm** em uma prova, e seu professor provavelmente achará também.

Veja um exemplo de uma prova onde a **Comm** é útil:

$P \wedge (\sim Q \rightarrow R) : (R \vee Q) \wedge P$

1.	$P \wedge (\sim Q \rightarrow R)$	**P**
2.	P	1 **Simp**
3.	$\sim Q \rightarrow R$	1 **Simp**
4.	$Q \vee R$	3 **Impl**

Aqui aparece a oportunidade para você usar a **Comm** — não vá deixá-la passar!

5.	$R \vee Q$	4 **Comm**
6.	$(R \vee Q) \wedge P$	5 **Conj**

Associação (Assoc)

A **Assoc** lhe diz que nas proposições em que todos operadores são $\wedge$ ou $\vee$, você pode mover os parênteses livremente.

Assoc: $(x \wedge y) \wedge z :: x \wedge (y \wedge z)$

$(x \vee y) \vee z :: x \vee (y \vee z)$

Como na **Comm**, a **Assoc** tem uma prima na Aritmética. Por exemplo, (3 + 4) + 5 = 3 + (4 + 5)

A **Assoc** e a **Comm** podem ser poderosas ferramentas quando usadas juntas. Usando apenas essas duas regras você pode rearranjar, da maneira que quiser, qualquer proposição que consista inteiramente de proposições $\wedge$ ou $\vee$. Apenas tome cuidado para fazer isso passo a passo.

Experimente esta prova:

$(P \rightarrow Q) \vee R : Q \vee (R \vee \sim P)$

Como sempre, escreva as premissas primeiro:

1. $(P \rightarrow Q) \vee R$ **P**

Observe que a conclusão tem apenas proposições $\vee$. Assim, se você puder encontrar uma maneira de escrever a premissa usando somente proposições $\vee$, poderá concluir a prova usando somente a **Comm** e a **Assoc**. Veja o passo seguinte:

2. $(\sim P \vee Q) \vee R$ 1 **Impl**

Observe que usei a **Impl** somente em *parte* da premissa.

Nesse ponto, a estratégia é apenas rearranjar as constantes na linha 2 e fazer com que essa proposição pareça com a conclusão. Já que a primeira constante na conclusão é *Q*, o próximo passo colocará o *Q* na posição apropriada:

3. $(Q \vee \sim P) \vee R$ 2 **Comm**

Observe que novamente apliquei a regra de equivalência para parte da proposição. Depois vou mover os parênteses para a direita usando a **Assoc**:

4. $Q \vee (\sim P \vee R)$ 3 **Assoc**

Tudo o que restou é inverter *~R* e *P*:

5. $Q \vee (R \vee \sim P)$ 4. **Comm**

Essa prova em particular também funciona ao contrário, isto é, você pode usar a conclusão para provar a premissa. Isso acontece em *qualquer* prova que tenha somente uma premissa e usa somente regras de equivalência. Isso também lhe diz que a premissa e a conclusão são semanticamente equivalentes.

Distribuição (Dist)

Como na **Exp**, a **Dist** é um tanto confusa até que você a coloque em palavras e, aí, ela passa a fazer sentido.

Dist: $x \wedge (y \vee z) :: (x \wedge y) \vee (x \wedge z)$

$x \vee (y \wedge z) :: (x \vee y) \wedge (x \vee z)$

Essa regra também tem uma análoga na Aritmética. Por exemplo:

$2 \times (3+5) = (2 \times 3) + (2 \times 5)$

Por essa razão, diz-se que a multiplicação se *distribui* sobre a adição. Observe que seu inverso não é verdadeiro:

$2 + (3 \times 5) \neq (2 + 3) \times (2 + 5)$

Na LS, porém, os operadores $\wedge$ e $\vee$ se distribuem um sobre o outro. Veja como funciona. Comece com uma proposição em português que se encaixe no formato $x \wedge (y \vee z)$:

Eu tenho um animal de estimação *e* ele é um cachorro *ou* um gato.

Esta é a proposição paralela para $(x \wedge y) \vee (x \wedge z)$:

Ou eu tenho um animal de estimação *e* é um gato *ou* tenho um animal de estimação *e* é um cachorro.

Essas duas proposições significam a mesma coisa, o que pode ajudá-lo a entender por que a **Dist** funciona.

Da mesma forma, veja como o operador $\vee$ se distribui sobre o operador $\wedge$. Desta vez comece com uma proposição que se enquadre no formato $x \vee (y \wedge z)$:

Eu tenho que ter aulas de Química Orgânica *ou de ambas*, Botânica *e* Zoologia.

A proposição paralela para $(x \vee y) \wedge (x \vee z)$ é um pouco complicada de traduzir, mas veja:

Eu tenho que ter aulas *ou* de Química Orgânica *ou* de Botânica *e* também tenho que ter aulas *ou* de Química Orgânica *ou* de Zoologia.

Compare essa proposição com a anterior para se convencer de que elas querem dizer a mesma coisa.

Veja um exemplo de prova para começar:

$Q \vee R$, $\sim(P \wedge Q)$, $P : R$

1.	$Q \vee R$	**P**
2.	$\sim(P \wedge Q)$	**P**
3.	P	**P**

Você poderia aplicar **Impl** e **Contra** na proposição 1, não faria mal algum, e essa tática é recomendável quando está buscando por ideias de onde começar. No entanto, tomarei outro caminho para usar a **Dist**:

4.	$P \wedge (Q \vee R)$	1, 3 **Conj**

5. $(P \wedge Q) \vee (P \wedge R)$ 4 **Dist**

Agora, o jogo já começou. O que vem depois?

Quando ficar empacado no meio de uma prova, analise as premissas que ainda não usou para buscar novas ideias.

Observe que a segunda premissa é apenas a negação da primeira parte da proposição 5, o que o permite usar **SD**:

6. $P \wedge R$ 2,5 **SD**

7. R 6 **Simp**

Essa é uma provinha complicada, e você pode estar se perguntando o que teria feito se não soubesse usar a **Conj** para criar a proposição 4. No próximo passo, mostrarei como elaborar essa prova de uma maneira inteiramente diferente.

Teorema de DeMorgan (DeM)

Assim como a **Dist** e a **Exp**, a **DeM** também se torna mais clara quando você usa proposições em português.

DeM: $\sim(x \wedge y) :: \sim x \vee \sim y$

$\sim(x \vee y) :: \sim x \wedge \sim y$

Veja uma proposição em português que se encaixe no formato $\sim(x \wedge y)$. Observe que essa é uma proposição do tipo *não....ambos* que você viu no Capítulo 4:

Não é *verdade* que sou *ambos:* rico *e* famoso.

Eis a proposição paralela que se encaixa no formato $\sim x \vee \sim y$:

Ou eu *não* sou rico *ou não* sou famoso.

Como você pode ver, essas duas proposições significam a mesma coisa, o que gera um entendimento intuitivo da razão pela qual suas formas correspondentes são equivalentes.

Veja a proposição que se encaixa na forma $\sim(x \vee y)$. Note que esta é uma situação de *não...e não*, que trato no Capítulo 4.

> *Não é* verdade que Jack é médico *ou* advogado.

A proposição paralela que se encaixa no formato $\sim x \wedge \sim y$ *é*:

> Jack *não é* médico *e não é* advogado.

Use a **DeM** para trocar as proposições dos formatos $\sim(x \wedge y)$ *e* $\sim(x \vee y)$ por formas mais fáceis de trabalhar.

Veja o argumento usado no tópico anterior:

> $Q \vee R, \sim(P \wedge Q), P : R$

1.	$Q \vee R$	**P**
2.	$\sim(P \wedge Q)$	**P**
3.	P	**P**

Dessa vez, ao invés de preparar tudo para usar a **Dist**, começo pela aplicação da **DeM** na proposição 2:

4.	$\sim P \vee \sim Q$	2 **DeM**

Sendo mais fácil de trabalhar com essa proposição, é mais provável que você siga este passo:

5.	$\sim Q$	3, 4 **SD**

A prova agora termina praticamente sozinha:

6.	R	1, 5 **SD**

Há sempre mais de um caminho para se seguir em uma prova. Se sua primeira tentativa não funcionar, tente outra rota.

Tautologia (Taut)

A **Taut** é a regra mais fácil deste capítulo.

Taut: $x \wedge x :: x$

$x \vee x :: x$

Na verdade é quase ridícula de tão fácil, por que se preocupar?

Para ser sincero, você não precisa da **Taut** em proposições $\wedge$. Pode trocar $x \wedge x$ para x, usando a **Simp,** e trocar x por $x \wedge x$, usando a **Conj**. Da mesma forma, no caso de proposições $\vee$, você pode trocar x por $x \vee x$, usando a **Add**. No entanto, se acabar com $x \vee x$ e precisar do x, a **Taut** é o melhor caminho.

Mesmo assim, só consigo pensar em uma maneira de acabar nessa situação, e é até bem simpática:

$P \rightarrow \sim P : \sim P$

Posso até ouvir você perguntando: “Essa prova é possível?” Sim, é. Observe:

1.	$P \rightarrow \sim P$	**P**
2.	$\sim P \vee \sim P$	1 **Impl**
3.	$\sim P$	2 **Taut**

Assim, assumindo que $P \rightarrow \sim P$ é verdadeira, você pode provar que $\sim P$ também é verdadeira.

Equivalência (Equiv)

Você pode estar pensando que esqueci das proposições $\leftrightarrow$. Na verdade, vai perceber que das 18 regras discutidas neste capítulo e no Capítulo 9, a **Equiv** é a única que inclui proposições $\leftrightarrow$.

Equiv: $x \leftrightarrow y :: (x \rightarrow y) \wedge (y \rightarrow x)$

$x \leftrightarrow y :: (x \wedge y) \vee (\sim x \wedge \hat{y})$

Agora a verdade pode ser contada: ignorei essas proposições até agora porque são absolutamente intratáveis quando trabalhadas em provas.

Um dos motivos para a rebeldia das proposições $\leftrightarrow$ é que suas tabelas verdade são perfeitamente simétricas. Ao contrário das tabelas dos outros três operadores binários, as do operador $\leftrightarrow$ se dividem igualmente em duas proposições verdadeiras e duas falsas. Essa simetria é muito bonita esteticamente, mas não ajuda em nada quando você tenta reduzir o campo para apenas *uma* possibilidade.

Assim, quando um argumento contém um operador $\leftrightarrow$, você tem que se livrar dele, e quanto antes, melhor. Sua sorte é que as duas regras **Equiv** podem ajudá-lo a fazer isso.

A primeira das duas formas de **Equiv** explora a ideia de que em uma proposição $\leftrightarrow$, as setas apontam para as duas direções. Isso faz com que possam ser divididas em duas proposições $\rightarrow$, interligadas por um operador $\wedge$. A segunda forma de **Equiv** explora a ideia de que ambas as partes de uma proposição $\leftrightarrow$ têm o mesmo valor verdade, o que significa que tanto *x* quanto *y* são verdadeiras ou falsas.

Observe o argumento a seguir:

$P \leftrightarrow (Q \wedge R), Q : P \vee \sim R$

1. $P \leftrightarrow (Q \wedge R)$ **P**

2.	Q	**P**

DICA

Quando tiver uma proposição $\leftrightarrow$ como premissa, automaticamente escreva ambas as formas de **Equiv** em sua prova:

3.	$(P \to (Q \wedge R)) \wedge ((Q \wedge R) \to P$	1 **Equiv**
4.	$(P \wedge (Q \wedge R)) \vee (\sim P \wedge \sim(Q \wedge R))$	1 **Equiv**

Enquanto faz isso, use a **Simp** para repartir a primeira versão de **Equiv** da proposição ^:

5.	$P \to (Q \wedge R)$	3 **Simp**
6.	$(Q \wedge R) \to P$	3 **Simp**

Voilà! Você escreveu quatro proposições automaticamente. Nesse ponto, analise o que tem para trabalhar. Nesse caso, a linha 6 parece uma **Exp** pronta para acontecer:

7.	$Q \to (R \to P)$	6 **Exp**

A partir daqui, observe que ainda não tocou na linha 2, e tudo vai para seu devido lugar, assim:

8.	$R \to P$	2,7 **MP**
9.	$\sim R \vee P$	8 **Impl**
10.	$P \vee \sim R$	9 **Comm**

Nesse caso você não precisará da segunda forma de **Equiv**. Mas veja a seguinte prova onde ela é necessária:

$P \leftrightarrow Q, R \to (P \vee Q) : R \to (P \wedge Q)$

Essa é a prova mais difícil deste capítulo.

Estude todos os passos para ter certeza de que os entendeu e tente reproduzi-la com o livro fechado. Se você dominar essa regra, terá um bom entendimento de uma das mais difíceis das 18 regras de inferência.

Como sempre, as premissas vêm primeiro:

1.	$P \leftrightarrow Q$	**P**
2.	$R \leftrightarrow (P \vee Q)$	**P**

Aqui estão todas as quatro proposições que você pode escrever sem muito esforço, todas da primeira premissa:

3.	$(P \rightarrow Q) \wedge (Q \rightarrow P)$	1 **Equiv**
4.	$(P \wedge Q) \vee (\sim P \wedge \sim Q)$	1 **Equiv**
5.	$P \rightarrow Q$	3 **Simp**
6.	$Q \rightarrow P$	3 **Simp**

Agora, observe que a linha 2 é $R \rightarrow (P \vee Q)$ e a conclusão é $R \rightarrow (P \wedge Q)$. Então, se encontrar uma forma de criar $(P \vee Q) \rightarrow (P \wedge Q)$, a regra de implicação **SH** lhe dará a conclusão.

No final, porém, somente a linha 4 será útil e permitirá o seguinte caminho:

7.	$\sim(P \wedge Q) \rightarrow (\sim P \wedge \sim Q)$	4 **Impl**
8.	$\sim(\sim P \wedge \sim Q) \rightarrow (P \wedge Q)$	7 **Contra**
9.	$(P \vee Q) \rightarrow (P \wedge Q)$	8 **DeM**
10.	$R \rightarrow (P \wedge Q)$	2, 6 **SH**

Capítulo 11

Grandes Suposições com Provas Indiretas e Condicionais

Neste Capítulo

- Entendendo a prova condicional
- Presumindo e descartando premissas
- Provando argumentos com a prova indireta
- Misturando a prova condicional com a indireta

Você já deve ter visto aqueles horríveis comerciais de TV em que o anunciante fica perguntando "*Agora*, quanto você pagaria?", à medida que acrescenta mais coisas que você não quer a seus óculos de proteção de $19,95 – o ralador de queijo, o picador de gelo e o descascador de batatas que escreve debaixo da água. Este capítulo será um pouco semelhante a esses comerciais, mas com uma diferença fundamental: vou acrescentar uma porção de coisas que você *realmente* quer.

Neste capítulo, apresentarei duas novas formas de prova: a *condicional* e a *indireta*, completamente grátis. Ao contrário das *provas diretas* que mostrei nos Capítulos 9 e 10, as condicionais e indiretas envolvem uma *suposição*, que é uma premissa adicional que você não *sabe* se é verdadeira, mas *presume* que sim.

Ainda mostrarei, neste capítulo e igualmente sem nenhum custo, não só como, mas também quando usar cada um desses métodos.

LEMBRE-SE

A prova condicional quase sempre torna a prova mais fácil, mas nem sempre pode ser usada. No entanto, quando a conclusão de um argumento é uma proposição do tipo →, a prova condicional é geralmente a melhor opção. Por outro lado, a indireta, também chamada de *prova de contradição*, é um método de escala industrial que funciona para toda prova. Entretanto, mesmo que seja algumas vezes o *único* método que funciona, não é necessariamente o caminho *mais fácil*, de forma que você deve usá-lo com economia.

Condicionando Suas Premissas com as Provas Condicionais

Você tem sido um estudante aplicado: praticou as provas ao ponto de conseguir fazê-las dormindo. Decorou todas as regras de implicação e equivalência dos Capítulos 9 e 10. Parabéns, você sobreviveu ao inferno!

Você pensa que está preparado para ser o consultor de Lógica de sua escola e, aí, este argumento aparece para abalar sua confiança:

$$P \rightarrow \sim Q,\ P \vee \sim R : Q \rightarrow ((\sim P \wedge \sim R) \vee (Q \rightarrow P))$$

Bem, pelo menos você sabe por onde começar:

1.	$P \rightarrow \sim Q$	**P**
2.	$P \vee \sim R$	**P**

Bom, depois de escrever as premissas, não há dúvidas de que essa será uma prova complicada.

Mas, qual o mal de tentar a velha tática? Veja algumas das proposições que você pode criar:

3.	$Q \rightarrow \sim P$	1 **Contra**
4.	$\sim Q \vee \sim P$	1 **Impl**
5.	$\sim(Q \wedge P)$	4 **DeM**
6.	$\sim(\sim P \wedge R)$	2 **DeM**
7.	$\sim P \rightarrow \sim R$	2 **Impl**
8.	$R \rightarrow P$	7 **Contra**
9.	$R \rightarrow \sim Q$	1, 8 **SH**
10.	$Q \rightarrow R$	9 **Contra**

11. $\sim Q \vee R$ — 10 **Impl**

Nenhuma dessas proposições realmente o leva aonde você precisa ir. Dessa forma, é hora de adotar uma nova tática – fácil de usar e que realmente funcione. Sim, você adivinhou: essa tática é a prova condicional.

Entendendo a prova condicional

A prova condicional lhe permite usar parte da conclusão como uma premissa para provar o restante da conclusão.

LEMBRE-SE

Para provar a validade de um argumento cuja conclusão está no formato $x \rightarrow y$ (isto é, qualquer proposição $\rightarrow$), você pode seguir estes passos:

1. Destaque a subproposição *x.*
2. Adicione *x* à sua lista de premissas como *Premissa Presumida* (PP).
3. Prove a subproposição *y* como se ela fosse a conclusão.

A ideia é simples, mas brilhante. Assim, no exemplo do tópico anterior, ao invés de procurar um caminho e tentar provar que a conclusão $Q \rightarrow ((\sim P \wedge \sim R) \vee (Q \rightarrow P)$ é válida, a prova condicional lhe permite:

1. Destacar a subproposição *Q.*
2. Adicionar *Q* à lista de premissas como uma **PP**.
3. Provar a subproposição $(\sim P \wedge \sim R) \vee (Q \rightarrow P)$, como se ela fosse a conclusão.

LEMBRE-SE

As premissas são as mesmas, mas agora você tem uma *Premissa Presumida* adicional (**PP**). Veja como funciona:

1. $P \rightarrow \sim Q$ — **P**

2.	$P \vee {\sim}R$	**P**
3.	Q	**PP**

Com essa prova, você está tentando construir a proposição $(({\sim}P \wedge {\sim}R) \vee (Q \rightarrow P))$. Desta vez, portanto, há um jeito de desmembrar as premissas e obter as partes de que precisa. Por exemplo, veja estes passos:

4.	${\sim}P$	1,3 **MT**
5.	${\sim}R$	2,4 **SD**

Com essas três premissas à mão, você pode trabalhar:

6.	${\sim}P \wedge {\sim}R$	4,5 **Conj**
7.	$({\sim}P \wedge {\sim}R) \vee (Q \rightarrow P)$	6 **Add**

Para concluir a prova, este é o último passo:

8.	$Q \rightarrow (({\sim}P \wedge {\sim}R) \vee (Q \rightarrow P))$	3-7 **PC**

Este passo final é chamado de *descarte da **PP***, que deixa claro que mesmo tendo feito a prova *como se* a suposição fosse verdadeira, nas linhas 3 a 7, você não está mais fazendo essa suposição na proposição 8. Em outras palavras, a conclusão é verdadeira, mesmo que a presunção *não seja*!

Você pode estar se perguntando: “Mas, não posso fazer isso, posso? Eu não provei que a conclusão *real* é verdadeira. Tudo que provei é que *parte* dela é verdadeira. Pior que isso, provei usando uma premissa falsa roubada da conclusão”.

Você está certo. Essa configuração parece boa demais para ser verdade. Afinal de contas, premissas são como dinheiro no banco, e a conclusão é como uma enorme dívida no cartão de crédito que você prefere nem olhar. Mas, e se eu lhe dissesse que poderia reduzir sua dívida pela metade *e* guardar dinheiro no banco? Isso é exatamente o que a prova condicional lhe permite fazer – é justo, é legal e nenhuma agência de cobrança virá atrás de você.

Por exemplo, lembre-se de que o argumento original era assim:

$$P \to \sim Q, \sim(\sim P \wedge R) : Q \to (\sim(P \vee R) \vee (Q \to P))$$

Se a conclusão pudesse falar, diria: “Você precisa me mostrar que se Q é verdadeira, então $\sim(P \vee R) \vee (Q \to P)$ também é verdadeira”.

Usando a prova condicional, você pode responder para a conclusão, dizendo: “Certo, então mostrarei a você que presumindo que Q é verdadeira $\sim(P \vee R) \vee (Q \to P)$ também é verdadeira”. E aí faz exatamente isto: *presume* que Q é verdadeira e depois prova o resto.

Adaptando a conclusão

Você pode aplicar regras de equivalência na conclusão de um argumento para tornar a prova condicional mais fácil de usar.

Invertendo as conclusões com a Contra

Para extrair todas as vantagens trazidas por este tópico, você tem que lembrar, primeiro, que pode usar todas as proposições → de duas formas: a forma em que se apresenta e sua **contraposição** (veja o Capítulo 10). Então, quando a conclusão é uma proposição →, você pode usar a prova condicional para atacá-la de duas diferentes maneiras.

Por exemplo, veja esta prova, feita com a forma original da proposição:

$$P \to Q, R \vee (Q \to P) : \sim(P \leftrightarrow Q) \to R$$

1.	$P \to Q$	**P**
2.	$R \vee (Q \to P)$	**P**

3. $\sim(P \leftrightarrow Q)$ **PP**

Se começar a prova dessa maneira, a Premissa Presumida não será muito útil.

Mas, imagine se você usar a **contraposição** na conclusão para obter $\sim R \rightarrow (P \leftrightarrow Q)$. Usar a **contraposição** lhe permite seguir os seguinte passos:

1.	$P \rightarrow Q$	**P**
2.	$R \vee (Q \rightarrow P)$	**P**
3.	$\sim R$	**PP**

Nesse caso, você está tentando provar que $(P \leftrightarrow Q)$. Esta solução é muito mais direta:

4.	$Q \rightarrow P$	2, 3 **SD**
5.	$(P \rightarrow Q) \wedge (Q \rightarrow P)$	1,5 **Conj**
6.	$P \leftrightarrow Q$	5 **Equiv**

Agora, você pode descartar a **PP**, da seguinte forma:

7.	$\sim R \rightarrow (P \leftrightarrow Q)$	3-6 **PC**

Não se esqueça de chegar até a conclusão:

8.	$\sim(P \leftrightarrow Q) \rightarrow R$	7 **Contra**

Quaisquer alterações que você fizer na conclusão aparecerão no final da prova, mesmo depois de descartar a suposição.

Nesse caso, mesmo que você pense em usar a **contraposição** primeiro e escrever a prova tendo isso em mente, na verdade ela aparecerá por último.

Vencendo com a implicação

Você pode transformar qualquer proposição $\vee$ em $\rightarrow$ usando a **implicação** (veja o Capítulo 10). Usar a **Impl** torna qualquer proposição $\vee$ uma candidata em potencial para a prova condicional.

Por exemplo, considere este argumento:

$$P : \sim R \vee (Q \rightarrow (P \wedge R))$$

Sem a prova condicional você não tem muita esperança. Mas o problema fica muito mais simples depois que você perceber que pode usar a **Impl** para reescrever a conclusão como

$$R \rightarrow (Q \rightarrow (P \wedge R))$$

Melhor ainda, você pode usar a **Exp** para reescrevê-la novamente:

$$(R \wedge Q) \rightarrow (P \wedge R)$$

Agora você está pronto para fazer a prova:

1.	P	**P**
2.	$R \wedge Q$	**PP**

Daí você quer extrair $P \wedge R$. Estes passos praticamente se escrevem sozinhos:

3.	R	2 **Simp**
4.	$P \wedge R$	1,3 **Conj**

Você está pronto, quase sem esforço, para descartar a **PP**:

5. $(R \wedge Q) \rightarrow (P \wedge R)$ — 2-4 **PC**

Depois que você descartar a **PP**, basta caminhar de volta para a forma original da conclusão:

6. $R \rightarrow (Q \rightarrow (P \wedge R))$ — 5 **Exp**
7. $\sim R \vee (Q \rightarrow (P \wedge R))$ — 6 **Impl**

Acrescentando mais de uma suposição

Depois que você assume uma premissa, se a nova conclusão for uma proposição → (ou puder ser transformada em uma), você pode assumir outra premissa. Essa é uma boa maneira de obter duas (ou mais!) presunções pelo preço de uma. Veja o exemplo:

$\sim Q \vee R : (P \vee R) \rightarrow ((Q \wedge S) \rightarrow (R \wedge S))$

Comece, como sempre, com as premissas e a **PP**:

1. $\sim Q \vee R$ — **P**
2. $P \vee R$ — **PP**

Infelizmente, você ainda tem um longo caminho para provar $(Q \wedge S) \rightarrow (R \wedge S)$. Mas, já que a nova conclusão é uma proposição →, pode extrair outra **PP**, da seguinte forma:

3. $Q \wedge S$ — **PP**

Agora, o novo objetivo é provar $R \wedge S$. Veja como são esses passos:

4. Q — 3 **Simp**

5.	*S*	3 **Simp**
6.	*R*	1, 4 **SD**
7.	$R \wedge S$	5, 6 **Conj**

Neste ponto, você pode descartar a última **PP** que presumiu:

8.	$(Q \wedge S) \rightarrow (R \wedge S)$	3-7 **PC**

Agora, pode descartar a primeira **PP**:

9.	$(P \vee R) \rightarrow ((Q \wedge S) \rightarrow (R \wedge S))$	2-8 **PC**

Quando você assume mais do que uma premissa, pode descartá-las na ordem inversa: *primeiro* a *última* premissa e, depois, na ordem inversa até a primeira.

Pensando Indiretamente: Provando Argumentos com a Prova Indireta

Quando você pensava que tudo estava indo bem, se depara com um argumento que simplesmente não consegue desvendar. Considere este argumento:

$P \rightarrow (Q \wedge \sim R), R : \sim(P \vee \sim R)$

1.	$P \rightarrow (Q \wedge \sim R)$	**P**
2.	R	**P**

Parece que não há muito o que fazer com esse argumento, não é? E já que a conclusão está em um formato que você não pode transformar em uma proposição →, a prova condicional está fora de questão, de modo que você pode pensar que não tem saída. A boa notícia é que não é verdade. De fato, um novo mundo está prestes a se abrir; e é um belo mundo!

Este tópico lhe mostra como usar a *prova indireta* que, diferentemente da condicional, é *sempre* uma opção, não importa qual seja a aparência da conclusão.

Apresentando a prova indireta

A prova indireta (também chamada de *prova por contradição*) é um tipo de judô lógico. A ideia aqui é presumir que a conclusão é falsa e, então, provar por que essa presunção está errada. Quando você consegue, significa que a conclusão é verdadeira.

LEMBRE-SE

Para provar que qualquer argumento é válido, você pode seguir os seguintes passos:

1. Negar a conclusão.

2. Adicionar a negação à sua lista de premissas como uma presunção.
3. Provar qualquer *proposição contraditória* (qualquer proposição do formato $x \wedge \sim x$).

Quando trabalhar no exemplo anterior deste tópico, ao invés de tentar provar a conclusão $\sim(P \vee \sim R)$, use a prova indireta, que permite utilizar sua negação, $P \vee \sim R$, como uma Premissa Presumida (**PP**). Lembre-se, apenas, de que agora você está tentando provar que essa presunção leva a uma proposição contraditória.

Veja como a prova deve ser:

$P \to (Q \wedge \sim R), R : \sim(P \vee \sim R)$

1.	$P \to (Q \wedge \sim R)$	**P**
2.	R	**P**
3.	$P \vee \sim R$	**PP**

LEMBRE-SE

Seu objetivo agora é provar a proposição e sua negação, então use-as para construir uma proposição ∧ contraditória, assim:

Com a **PP** à sua disposição, as opções se abrem:

4.	P	2,3 **SD**
5.	$Q \wedge \sim R$	1, 4 **MP**
6.	$\sim R$	**Simp**

Nesse ponto, você desenvolveu tanto R quanto $\sim R$, de forma que pode construí-las em uma única proposição contraditória, da seguinte forma:

7.	$R \wedge \sim R$	2, 6 **Conj**

A suposição levou-o a uma situação impossível, pois sabe que a **PP** deve ser falsa. Se $P \vee {\sim}R$ é falsa, então ${\sim}(P \vee {\sim}R)$ deve ser verdadeira:

8.	${\sim}(P \vee {\sim}R)$	3 -7 **PI**

Assim como na prova condicional, você precisa descartar a **PP**, o que deixa claro que mesmo que esteja operando como se a suposição fosse verdadeira a partir das proposições 3 e 7, não está mais fazendo essa suposição na proposição 8. Na verdade, a única coisa que você provou é que a suposição *não é* verdade!

Provando conclusões curtas

Como já mencionei no Capítulo 9, os argumentos em que a conclusão é mais curta que as premissas tendem a ser mais difíceis de provar do que aqueles em que as conclusões são mais longas, já que desmembrar premissas longas pode ser complicado.

Entretanto, a prova indireta funciona especialmente bem quando a conclusão é mais curta do que as premissas, porque a conclusão negada se torna uma boa premissa curta para ser usada. Por exemplo, considere este argumento:

$${\sim}(({\sim}P \vee Q) \wedge R) \rightarrow S,\ P \vee {\sim}R : S$$

1.	${\sim}(({\sim}P \vee Q) \wedge R) \rightarrow S$	**P**
2.	$P \vee {\sim}R$	**P**
3.	${\sim}Q \vee S$	**P**

De um jeito ou de outro, você terá que desmembrar a primeira premissa, mas será mais fácil com alguma ajuda:

4.	${\sim}S$	**PP**

Caminhos começam a se abrir imediatamente, e podemos seguir os seguintes passos:

5.	$(\sim P \vee Q) \wedge R$	1,4 **MT**
6.	$\sim P \vee Q$	5 **Simp**
7.	R	5 **Simp**

Lembre-se, você está tentando originar duas proposições contraditórias. Mas, agora, as oportunidades são mais abundantes:

8.	P	2,7 **SD**
9.	Q	6,8 **SD**
10.	S	3,9 **SD**

Quando estiver fazendo uma prova indireta, não cometa o engano de pensar que a prova está terminada depois que tiver provado a conclusão. Lembre-se de que ainda precisará construir a proposição contraditória.

Neste caso, a **PP** leva à sua própria negação, o que o permite concluir a prova:

11.	$S \wedge \sim S$	4, 10 **Con**
12.	S	4-11 **PC**

Aparentemente, a linha 12 se parece com a 10, mas, agora que você descartou a **PP**, a prova está concluída.

Combinando a Prova Condicional e a Indireta

Uma questão bastante comum entre estudantes de Lógica é: “Se eu já estou usando a **PP** para a prova condicional, tenho que começar tudo de novo se quiser adicionar uma **PP** para a prova indireta?”.

A boa notícia é que você não tem que recomeçar, veja no próximo exemplo:

$\sim P \wedge Q \rightarrow (\sim R \wedge S), Q : R \rightarrow P$

1.	$\sim P \wedge Q \rightarrow (\sim R \wedge S)$	**P**
2.	Q	**P**

Na primeira passagem da prova, você só consegue fazer isto:

3.	$\sim(\sim P \wedge Q) \vee (\sim R \wedge S)$	1 **Impl**

Talvez o próximo passo esteja aí, mas, de qualquer forma, você não está conseguindo enxergá-lo. Tem que seguir adiante. Já que a prova condicional é uma opção, tente fazê-la primeiro:

4.	R	**PP** (para a prova condicio

Agora, você está tentando provar *P*, mas ainda não tem certeza de como fazêlo diretamente. Pode passar para a prova indireta pela negação daquilo que está tentando provar e sua adição como uma das premissas:

5.	$\sim P$	**PP** (para a prova indireta

O objetivo agora é encontrar uma contradição. De repente as peças se encaixam em seus lugares:

6.	$\sim P \wedge Q$	2, 5 **Conj**
7.	$\sim R \wedge S$	1, 6 **MP**
8.	$\sim R$	7 **Simp**

Agora você está pronto para descartar a **PP** para a prova indireta:

9.	$R \wedge \sim R$	4,8 **Conj**
10.	P	5-9 **PI**

É claro que provar *P* era o objetivo da prova condicional original, assim, aqui está o que você obtém:

11.	$R \rightarrow P$	4- 10 **PC**

LEMBRE-SE

Quando usar os métodos de prova condicional e indireta juntos, distribua as PPs começando pela última que adicionou e faça todo o processo até a primeira.

Capítulo 12

Juntando Tudo: Estratégias para Resolver Qualquer Prova

Neste Capítulo

- Lidando rapidamente com provas fáceis
- Usando prova condicional para lidar com provas moderadas
- Complicando a vida das provas difíceis

Alguns problemas de Lógica quase se resolvem sozinhos. Outros parecem difíceis à primeira vista, mas também se tornam fáceis quando você sabe o que fazer. Ainda há aqueles que lutam e esperneiam a cada passo, até que você consiga domá-los.

Este capítulo é todo sobre o que a Oração da Serenidade chama de "a sabedoria para saber a diferença". A sabedoria que conquistará aqui será a confiança serena de que você precisará para escrever provas na Lógica Sentencial (LS) com facilidade, quando possível, e perseverança, quando necessário.

Assim como nosso sistema jurídico declara que um réu é "inocente até que se prove o contrário", recomendo que a mesma abordagem seja adotada neste capítulo em relação às provas: "considere-as fáceis até que provem ser difíceis". Primeiro mostrarei como fazer uma rápida avaliação de um argumento para tentar sentir o grau de dificuldade da prova. Depois apresentarei alguns movimentos básicos e fáceis para lidar com as provas simples em cinco minutos ou menos.

Para provas mais teimosas, mostrarei como e quando usar a técnica da prova condicional vista no Capítulo 11. Essa técnica é, em geral, suficiente para concluir as provas de dificuldade média.

Finalmente, para aquelas realmente difíceis, mostrarei como usar uma das formas proposicionais da LS em benefício próprio, desmembrando as premissas longas e trabalhando a partir das extremidades em direção ao meio para concluir a prova. Também apresentarei um método mais avançado de prova indireta.

Provas Fáceis: Uma Abordagem Intuitiva

Você não precisa de uma metralhadora para matar um mosquito, nem de uma estratégia mirabolante para resolver um problema fácil. Basta gastar apenas cinco minutos analisando o que está à sua frente e escrevendo algumas ideias. Os tópicos a seguir mostram alguns truques para escrever provas simples com graça e rapidez.

Analise o problema

Analisar a proposição não significa apenas olhar. As três sugestões discutidas nos tópicos seguintes tomarão menos de um minuto, mas será um minuto precioso.

Compare as premissas e a conclusão

As premissas são parecidas com a conclusão ou são muito diferentes? Se forem parecidas, a prova pode não ser tão difícil; caso contrário, pode ser um pouco mais complicada.

Em ambos os casos, pense no que precisa acontecer para preencher essa lacuna. Você consegue pensar em alguma ideia para continuar?

Isso é tudo o que esse passo requer: pensar e trabalhar com seus instintos.

Observe o tamanho das premissas e da conclusão

De modo geral, premissas curtas e conclusão longa indicam uma prova fácil, enquanto que premissas longas e conclusão curta indicam provas mais difíceis.

Você pode construir qualquer conclusão quando tem premissas curtas suficientes. Por outro lado, desmembrar premissas longas até obter uma conclusão curta pode ser bem complicado.

No tópico seguinte, “Escreva o que é fácil”, mostrarei uma porção de maneiras para desmembrar as premissas. Por ora, apenas tente sentir o grau de dificuldade da prova com base nesta regra geral: *quanto mais curtas forem as premissas, mais fácil será a prova.*

Procure por trechos repetidos de proposições

Quando você notar que as proposições de um argumento possuem trechos repetidos, sublinhe-os para ressaltá-los. Nesse casso, a melhor estratégia é, de modo geral, deixar esses trechos de lado, ao invés de desmembrá-los.

Por exemplo, observe este argumento:

$$(P \leftrightarrow Q) \rightarrow \sim(R \wedge S),\ R \wedge S : \sim(P \leftrightarrow Q)$$

Você poderia “martelar” a proposição $(P \leftrightarrow Q) \rightarrow \sim(R \wedge S)$ até quebrá-la, mas não precisa disso. Depois de perceber que os trechos $(P \leftrightarrow Q)$ e $(R \wedge S)$ se repetem, a solução segue:

1.	$(P \leftrightarrow Q) \rightarrow \sim(R \wedge S)$	**P**
2.	$R \wedge S$	**P**
3.	$\sim(P \leftrightarrow Q)$	1, 2 **MT**

Por ora, a percepção dessas repetições já o manterá no caminho.

Escreva o que é fácil

Em apenas mais um minuto, este tópico mostrará mais quatro rápidas estratégias para continuar. Chamo esse tipo de estratégia de *x-tudo*, pois você usa nessa prova tudo a que tem direito.

Ao usar essas estratégias, se perceber que o caminho está livre, não hesite em ir até o final.

Desmembre as proposições usando a Simp e o SD

A **Simp** e o **SD** são as duas regras mais simples para desmembrar proposições $\land$ e $\lor$ (veja o Capítulo 9 pra saber tudo sobre a **Simp** e o **SD**). Quanto mais cedo você desmembrar as premissas, maiores serão as chances de chegar à conclusão.

Expanda suas opções usando a Impl e a Contra

Use a **Impl** (veja o Capítulo 10) para converter proposições $\lor$ em $\to$ e viceversa. Depois use a **Contra** para converter todas as proposições $\to$ em suas contraposições. Utilize essas regras para reescrever todas as proposições possíveis porque, em qualquer direção, essas são algumas maneiras fáceis de aumentar suas opções para mais tarde.

Use MP e MT sempre que possível

É muito fácil identificar as oportunidades para usar a **MP** e a **MT** (como visto no Capítulo 9), e elas tendem a resultar em proposições simples de serem trabalhadas.

Converta todas as proposições negativas usando a DeM

Falando de forma geral, a **DeM** é a única regra que permite converter as quatro formas negativas de proposições LS em positivas.

A **DeM** funciona diretamente nas proposições dos formatos $\sim(x \land y)$ e $\sim(x \lor y)$, como discutido no Capítulo 10. Mas, mesmo quando você tem que enfrentar proposições $\to$ e $\leftrightarrow$, que são as duas formas negativas restantes, pode usar a **DeM** depois de empregar algumas outras regras para transformar essas proposições nos formatos $\sim(x \land y)$ e $\sim(x \lor y)$.

Por exemplo, para converter $\sim(x \to y)$, siga os seguintes passos:

1.	$\sim(x \to y)$	**P**

2.	$\sim(\sim x \vee y)$	1 **Impl**
3.	$x \wedge \sim y$	2 **DeM**
4.	x	3 **Simp**
5.	$\sim y$	3 **Simp**

Em apenas alguns passos, você transformou uma proposição aparentemente complexa em duas simples.

Mesmo quando você está encurralado com a temida forma $\sim(x \leftrightarrow y)$, pode usar a **Equiv**, após desmembrá-la pedaço por pedaço com a **DeM**. Por exemplo, veja estes passos:

1.	$\sim(x \leftrightarrow y)$	**P**
2.	$\sim((x \wedge y) \vee (\sim x \wedge \sim y))$	1 **Equiv**
3.	$\sim(x \wedge y) \wedge \sim(\sim x \wedge \sim y)$	2 **DeM**
4.	$\sim(x \wedge y)$	3 **Simp**
5.	$\sim(\sim x \wedge \sim y)$	3 **Simp**

Nesse ponto, você usa a **DeM** novamente para simplificar as proposições 4 e 5 ainda mais. É claro que são alguns passos extras, mas você transformou uma proposição impenetrável em duas muito simples.

Saiba quando partir para outra

Você já gastou cinco minutos com um problema especialmente complicado. Virou e revirou sua cabeça. Já anotou algumas proposições simples – ou talvez não tenha nada para ser anotado, e, agora, esteja totalmente sem ideias.

Meu conselho: cinco minutos. Esse é todo o tempo que você deve gastar com a estratégia do instinto, podendo dobrá-lo se as premissas forem longas e a conclusão curta. Se a prova não surgir em sua frente em cinco minutos, precisará usar uma abordagem mais forte para que os cinco minutos seguintes sejam produtivos e não frustrantes.

DICA

Mesmo que você tenha que passar para uma nova tática, não tem que começar novamente. Qualquer proposição que já tenha provado é sua, e sintase à vontade para usá-la no resto da prova.

Provas Intermediárias: Sabendo Quando Usar a Prova Condicional

A abordagem mais forte será tratada neste tópico. Depois que tiver perdido a esperança de que o problema seja simples, é hora de lançar mão da prova condicional.

Use a prova condicional como sua primeira opção sempre que possível, pois ela tende a ser o caminho mais rápido para a solução do problema de dificuldade média.

Para decidir quando a prova condicional é possível, observe a conclusão que quer provar e decida em qual das oito formas básicas ela se encaixa (veja o Capítulo 5 para uma revisão das oito formas proposicionais básicas da LS).

Você sempre pode usar a prova condicional para três das oito formas básicas, as quais chamo de *formas amigáveis*. Pode, também, usá-la para as outras duas formas, que chamo de *formas menos amigáveis*, mas com um pouco mais de trabalho. Finalmente, você não pode usar a prova condicional nas três formas remanescentes, que chamo de *formas não amigáveis*:

Formas Amigáveis	***Formas menos Amigáveis***	***Formas não Amigáveis***
$x \to y$	$(x \leftrightarrow y)$	$\times \wedge y$
$x \vee y$	$\sim(x \leftrightarrow y)$	$\sim(x \to y)$
$\sim(x \wedge y)$	Disjunção	$\sim(x \vee y)$

Neste tópico, discuto os casos em que você pode usar a prova condicional. Guardarei os demais casos para o tópico “Provas Difíceis: Sabendo o que Fazer Quando as Coisas se Complicam”.

As três formas amigáveis: $x \to y$, $x \vee y$ e $\sim(x \wedge y)$

É óbvio que você sempre pode usar a prova condicional em qualquer conclusão no formato $x \to y$. Também pode usá-la em conclusões nos formatos $x \vee y$ e $\sim(x \wedge y)$.

Você pode transformar qualquer conclusão do formato $x \vee y$ em uma condicional, usando apenas a **Impl**. Essa regra a transforma em $\sim x \to y$.

Por exemplo, se quiser provar o seguinte argumento:

$R : \sim(P \wedge Q) \vee (Q \wedge R)$

1.	R	**P**

Não há nada acontecendo aqui. Mas, depois que você perceber que a conclusão é equivalente a $(P \wedge Q) \to (Q \wedge R)$, poderá usar a prova condicional:

2.	$P \wedge Q$	**PP**
3.	Q	2 **Simp**
4.	$Q \wedge R$	1, 3 **Conj**
5.	$(P \wedge Q) \to (Q \wedge R)$	2, 4 **PC**

Depois que sua PP é descartada, tudo o que lhe resta fazer é demonstrar como a proposição que acabou de construir é equivalente à conclusão que você está tentando provar:

6.	$\sim(P \wedge Q) \vee (Q \wedge R)$	5 **Impl**

Outra forma fácil de se trabalhar é $\sim(x \wedge y)$. Quando a conclusão estiver nesse formato, você precisará usar a **DeM** para tirá-la da forma negativa, mudando para $\sim x \vee$

$\sim y$. Desse ponto em diante, use a **Impl** para transformá-la em $x \rightarrow \sim y$.

Por exemplo, se quiser provar o seguinte argumento:

$\sim P \rightarrow Q,\ P \rightarrow \sim R : \sim(Q \wedge R)$

1.	$\sim P \rightarrow Q$	**P**
2.	$P \rightarrow \sim R$	**P**

A dica aqui é perceber que a conclusão é equivalente a $Q \vee \sim R$ (por **DeM**), que, aí, será equivalente a $\sim Q \rightarrow \sim R$ (por **Impl**). Novamente, você pode usar a prova condicional:

3.	$\sim Q$	**PP**
4.	P	1, 3 **MT**
5.	$\sim R$	2, 4 **MP**
6.	$\sim Q \rightarrow \sim R$	3-5 **PC**

Como nos exemplos anteriores, depois que você tiver descartado a PP, precisará concluir a ponte entre a proposição que construir e a conclusão com que tiver começado:

7.	$Q \vee \sim R$	6 **Impl**
8.	$\sim(\sim Q \wedge R)$	7 **DeM**

As duas formas menos amigáveis: $x \leftrightarrow y$ e $\sim(x \leftrightarrow y)$

Se já disse uma vez, repito mais mil vezes: trabalhar com proposições $\leftrightarrow$ é sempre um pouco arriscado. Mas, felizmente, o princípio aqui é o mesmo de quando se trabalha com as formas mais amigáveis.

> **DICA**
>
> Tenha sempre em mente, e por favor nunca se esqueça de que o primeiro passo em uma proposição $\leftrightarrow$, é sempre se livrar do operador $\leftrightarrow$, usando a **Equiv**. Lembrar esse pequeno conselho sempre o levará até a metade do caminho se acontecer de você se perder.

Para trabalhar com uma conclusão que não esteja na forma $x \leftrightarrow y$, você terá, primeiro, que usar a **Equiv** para transformá-la em $(x \land y) \lor (\sim x \land \hat{y})$. Deve reconhecer que uma proposição-$\lor$ é uma forma amigável, que lhe permite usar a **Impl** para substituí-la por $\sim(x \land y) \to (\sim x \land \sim y)$.

Por exemplo, se quiser provar o seguinte argumento:

$$((\sim P \lor Q) \lor \sim R) \to \sim(P \lor R) : P \leftrightarrow R$$

1.	$((\sim P \lor Q) \lor \sim R) \to \sim(P \lor R)$	**P**

Não vou enganá-lo: essa é uma prova difícil. Se não encontrar um caminho para transformá-la em condicional, você não tem grandes esperanças. Felizmente, pode usar a **Equiv** para mudar a conclusão para $(P \land R) \lor (\sim P \land \sim R)$ e, a partir daqui, substituí-la por $\sim(P \land R) \to (\sim P \land \sim R)$ usando a **Impl**. Veja:

2.	$\sim(P \land R)$	**PP**
3.	$\sim P \lor \sim R$	2 **DeM**

Agora, você está tentando provar $\sim P \land \sim R$. A grande questão nesse ponto é "como usar a premissa complicada da linha 1?" (essa é a razão pela qual digo que longas premissas fazem provas difíceis). Certo, uma coisa de cada vez: você pode ao menos desenrolar a premissa um pouquinho, usando a **DeM** na segunda parte:

4.	$((\sim P \lor Q) \lor \sim R) \to (\sim P \land \sim R)$	1 **DeM**

A segunda parte dessa proposição parece idêntica ao que você está tentando provar. Assim, se puder construir a primeira parte da proposição, você usará a **MP** para chegar à segunda parte. O objetivo, agora, é provar a proposição *(~P ∨ Q) ∨ ~R.*

Mas, agora, você deve estar se perguntando: como chegar até *Q* do nada? A grande percepção aqui é que você pode usar a **Add** para juntar *Q e ~P ∨ ~R:*

5.	*(~P ∨ ~R) ∨ Q*	3 **Add**

A próxima parte dessa prova requer apenas alguma manipulação usando a **Assoc** e a **Comm**:

6.	*~P ∨ (~R ∨ Q)*	5 **Assoc**
7.	*~P ∨ (Q ∨ ~R)*	6 **Comm**
8.	*(~P ∨ Q) ∨ ~R*	7 **Assoc**

Finalmente, você chega perto do Santo Graal: pode usar a linha 8 juntamente com a 4 para desenvolver o que precisa para descartar sua premissa:

9.	*~P ∧ ~R*	4, 8 **MP**
10.	*~(P ∧ R) → (~P ∧ ~R)*	2-9 **PC**

Como sempre, depois de descartar a **PP**, você precisa fazer com que a proposição que acabou de construir se pareça com a conclusão:

11.	*(P ∧ R) ∨ (~P ∧ ~R)*	10 **Impl**
12.	*P ↔ R*	11 **Equiv**

Certo, essa prova foi penosa. Mas transformar aquela conclusão difícil em uma mais maleável tornou-a totalmente possível.

Provar conclusões nos formatos menos amigáveis pode ficar complicado. Esse exemplo é certamente a definição do que chamo de dificuldade moderada. Mais adiante, darei um exemplo de como provar uma conclusão no formato $\sim(x \leftrightarrow y)$, no tópico “Provas Difíceis: Sabendo o que Fazer Quando as Coisas se Complicam”.

As três formas não amigáveis: $x \wedge y$, $\sim(x \vee y)$ e $\sim(x \rightarrow y)$

Quando a conclusão se encaixa nessa categoria, a prova condicional quase nunca é a opção, pois, via de regra, você não pode transformar essas formas em $x \rightarrow y$.

Para entender por que você não pode usar a prova condicional nesses casos, primeiro observe que não há regra para transformar proposição no formato $x \wedge y$ em uma do tipo $\rightarrow$. Da mesma maneira, as duas formas não amigáveis remanescentes são fáceis de serem transformadas em proposições $\wedge$, mas, novamente, você fica travado se tentar transformá-las em $\rightarrow$.

Sempre que encontrar uma conclusão em uma forma não amigável, meu conselho é seguir adiante e tentar qualquer das provas: direta ou indireta. Em qualquer dos casos, provavelmente será uma prova difícil.

Provas Difíceis: Sabendo o que Fazer Quando as Coisas se Complicam

Quando a tentativa instintiva de prova direta falha e você não pode usar a prova condicional, provavelmente está lidando com uma prova difícil. Nesses casos, terá que escolher entre tentar seguir adiante com a prova direta ou trocar para uma prova indireta.

Os tópicos a seguir trazem algumas estratégias para ajudar a enfrentar a batalha das provas realmente complicadas.

Escolha cuidadosamente entre a prova direta e a indireta

Se encontrar um problema em que a conclusão não possa ser transformada em uma proposição → (que lhe permitiria usar a prova condicional), considere a prova direta como a primeira opção, *exceto* quando encontrar uma das três exceções listadas neste tópico.

Mesmo quando você não encontrar uma prova direta nos primeiros cinco minutos, se não puder usar a prova condicional, continue com ela um pouco mais; na melhor das hipóteses, usando algumas das ideias trazidas mais adiante, neste capítulo, poderá encontrar uma prova direta sem precisar trocar para a indireta. Na pior hipótese, quando precisar trocar para a prova indireta, ainda poderá usar todas as proposições que tiver criado enquanto procurava pela prova direta.

Essa mudança de planos não é verdadeira na direção oposta: se você trocar para a prova indireta muito depressa e depois quiser abandonar o navio e tentar a prova direta novamente, não poderá usar qualquer das proposições que tiver encontrado enquanto procurava pela prova indireta.

Exceção 1: Quando a conclusão é curta

A prova indireta pode ser muito útil quando a conclusão é uma proposição curta e a maioria das (ou todas) premissas são longas. Nesse caso, transformar uma conclusão curta e enfadonha em uma premissa curta e útil é uma situação em que você só tem a ganhar.

Exceção 2: Quando a conclusão é longa, mas uma negação

A prova indireta é especialmente boa para lidar com conclusões em formas negativas – $\sim(x \vee y)$ e $\sim(x \rightarrow y)$ –, pois elas se transformarão em uma premissa positiva. Por exemplo:

$P, Q : \sim((Q \vee R) \rightarrow (\sim P \wedge R))$

1.	P	**P**
2.	Q	**P**
3.	$(Q \vee R) \rightarrow (\sim P \wedge R)$	**PP**

Usar a prova indireta transforma a conclusão em forma positiva e permite que ela seja usada como premissa. Agora, você pode continuar:

4.	$Q \vee R$	2 **Add**
5.	$\sim P \wedge R$	3, 4 **MP**
6.	$\sim P$	5 **Simp**
7.	$P \wedge \sim P$	1, 6 **Conj**
8.	$\sim((Q \vee R) \rightarrow (\sim P \wedge R))$	3-8 **PI**

Exceção 3: Quando tudo parecer perdido

A terceira hipótese para usar a prova indireta é quando você já cansou de bater a cabeça com a direta e não chegou a lugar algum. Apenas espere um pouco, é tudo o que lhe peço. Passar da prova direta para a indireta é sempre fácil. Mesmo que você já esteja usando a prova condicional, sempre poderá passar para a indireta apenas adicionando outra **PP** (veja o Capítulo 11 para um exemplo de como a **PP** funciona).

Trabalhe de trás para frente a partir da conclusão

Costumo comparar a elaboração de provas com a construção de uma ponte entre dois pontos. Em geral, é exatamente isso que acontece: você consegue atravessar daqui para lá. Mas, às vezes, isto não funciona assim tão bem. Quando estiver enrolado, sem ter para onde ir, pode ser mais fácil chegar *aqui* vindo de *lá.*

No Capítulo 9, mostrei um rápido exemplo de como trabalhar de trás para frente a partir da conclusão. Neste tópico, usarei esse método para escrever uma prova bem complicada. Veja o seguinte argumento:

$$P \to Q,\ (P \to R) \to S,\ (\sim Q \vee \sim S) \wedge (R \vee \sim S) : \sim(Q \leftrightarrow R)$$

A conclusão está no formato $\sim(x \leftrightarrow y)$, uma das duas proposições "menos amigáveis". Assim, se você quiser usar a prova condicional, terá que rearranjar a conclusão. Aqui, escrever o *final* da prova antes do começo é uma boa opção:

99. $\sim(Q \leftrightarrow R)$

Você começa numerando a última linha da prova como 99. Não precisa de tantas assim, mas pode renumerá-las no final. Como vê, a última linha contém a conclusão em toda a sua glória.

A partir daí, começa a trabalhar com a conclusão, para descobrir como chegou até ela. Nesse caso, o melhor caminho é usar a prova condicional, o que significa que a conclusão terá que ser uma proposição →. O primeiro passo é usar a **Equiv** para transformar a proposição na forma amigável $\sim(x \wedge y)$:

98.	$\sim((Q \to R) \wedge (R \to Q))$	
99.	$\sim(Q \leftrightarrow R)$	98 **Equiv**

Agora, você tem que decidir qual passo veio antes daquele que acabou de realizar. Desta vez, use a **DeM** para tirar a proposição 97 de sua forma negativa:

97. $\sim(Q \to R) \vee \sim(R \to Q)$

98.	$\sim((Q \to R) \wedge (R \to Q))$	97 **DeM**
99.	$\sim(Q \leftrightarrow R)$	98 **Equiv**

Depois que você pegar o jeito, verá que o próximo passo é transformar a proposição 97 em uma →, usando a **Impl**:

95.	$\sim(R \to Q)$	
96.	$(Q \to R) \to \sim(R \to Q)$	4-95 **PC**
97.	$\sim(Q \to R) \vee \sim(R \to Q)$	96 **Impl**
98.	$\sim((Q \to R) \wedge (R \to Q))$	97 **DeM**
99.	$\sim(Q \leftrightarrow R)$	98 **Equiv**

Depois de tudo isso, você tem o Santo Graal nas mãos. Sabe como a história termina: presumindo que $Q \to R$, e usando-a para construir $\sim(R \to Q)$, você descarta essa suposição juntando as duas subproposições, usando o operador → - .

Agora, é claro, sabe muito mais sobre como a história começa. Em especial, que sua **PP** deve ser $Q \to R$. De forma que os passos iniciais devem ser assim:

1.	$P \to Q$	**P**
2.	$(P \to R) \to S$	**P**
3.	$(\sim Q \vee \sim S) \wedge (R \vee \sim S)$	**P**
4.	$Q \to R$	**PP**

Agora, o objetivo é construir a proposição $\sim(R \to Q)$, que é o último passo que você descobriu no sentido oposto. Quando olha para a **PP** e para a linha 1, algo lhe salta aos olhos:

5.	$P \to R$	1, 4 **SH**

Então, olhe a linha 2 e perceba que a sorte está do seu lado:

6.	*S*	2, 5 **MP**

E agora? A linha 3 é a única premissa que resta e a forma parece ser bastante familiar:

7.	$(\sim Q \wedge R) \vee \sim S$	3 **Dist**

Aí você tem outro break:

8.	$\sim Q \wedge R$	6, 7 **SD**

Quando chega nesse ponto da prova, é recompensador saber manipular as oito formas básicas:

9.	$\sim(Q \vee \sim R)$	8 **DeM**
10.	$\sim(\sim R \vee Q)$	9 **Comm**
11.	$\sim(R \rightarrow Q)$	10 **Impl**

Nesse ponto, você alcançou seu objetivo e tudo o que precisa fazer é renumerar as últimas linhas:

12.	$(Q \rightarrow R) \rightarrow \sim(R \rightarrow Q)$	4-11 **PC**
13.	$\sim(Q \rightarrow R) \vee \sim(R \rightarrow Q)$	12 **Impl**
14.	$\sim((Q \rightarrow R) \wedge (R \rightarrow Q))$	13 **DeM**
15.	$\sim(Q \leftrightarrow R)$	14 **Equiv**

Vá fundo nas proposições da LS

A esta altura, você já deve ser um especialista em perceber em qual das oito formas a proposição se encaixa. Algumas vezes, porém, isso não é suficiente. Quando uma prova é difícil, geralmente depende do seu entendimento da estrutura das proposições em um nível mais profundo.

Você deve ter percebido que três das regras de equivalência desmembram a proposição em três partes (x, y e z), ao invés de apenas duas. Essas regras são a **Exp**, a **Assoc** e a **Dist** (veja o Capítulo 10). Você, provavelmente, não as usou tanto quanto as outras regras, mas precisará delas para as provas mais difíceis. Quando você começa a procurar oportunidades para usá-las, começa a encontrá-las.

Por exemplo, observe estes três argumentos:

$$(P \vee Q) \vee R$$

$$(P \wedge Q) \vee R$$

$$(P \wedge Q) \vee (R \vee S)$$

Você consegue visualizar que todas essas proposições têm a mesma forma básica $x \vee y$? Mas, não deixe que essa semelhança na estrutura básica impeça que você veja as importantes diferenças na estrutura mais profunda dessas proposições.

Por exemplo, você pode aplicar a **Assoc** para a primeira proposição, mas não na segunda; e a **Dist** na segunda, e não na primeira. Pode aplicar tanto a **Assoc** quanto a **Dist** na terceira proposição. Essas diferenças se tornam essenciais à medida que as provas se tornam mais difíceis. Leia este tópico para saber algumas dicas sobre como notar e explorar essas diferenças.

Usando a Exp

Exp: $x \to (y \to z)$ é equivalente a $(x \wedge y) \to z$

Por exemplo, veja este argumento:

$$(P \wedge Q) \to (R \to S)$$

Essa proposição está no formato $x \to y$. Mas, você também pode enxergá-la de duas outras formas. Uma possibilidade, considerando $(P \wedge Q)$ como uma só, é perceber que a proposição parece:

$$x \to (y \to z)$$

Nesse caso, você pode usar a **Exp** para transformar a proposição para:

$$((P \wedge Q) \wedge R) \to S$$

A segunda opção, considerando $(R \to S)$ como uma só, é perceber que a proposição parece com:

$$(x \wedge y) \to z$$

Desta vez, você pode usar a **Exp** na outra direção, assim:

$$P \to (Q \to (R \to S))$$

Combinando Assoc com Comm

Assoc: $(x \wedge y) \wedge z$ é equivalente a $x \wedge (y \wedge z)$

$(x \vee y) \vee z$ é equivalente a $x \vee (y \vee z)$

Por exemplo, considere o argumento:

$$\sim(P \vee Q) \to (R \vee S)$$

Um caminho aqui é aplicar a **Impl** para obter:

$$(P \vee Q) \vee (R \vee S)$$

Agora, você pode aplicar a **Assoc** em duas diferentes direções:

$$P \vee (Q \vee (R \vee S))$$

$$((P \vee Q) \vee R) \vee S$$

Pode, ainda, usar a **Comm** para rearranjar as variáveis em diversas maneiras diferentes. Por exemplo, apenas trabalhando com $P \vee (Q \vee (R \vee S))$, você obtém:

$$P \vee (Q \vee (S \vee R))$$

$$P \vee ((R \vee S) \vee Q)$$

$$(Q \vee (R \vee S)) \vee P$$

DICA

Se puder expressar uma proposição usando apenas operadores $\vee$ (ou somente operadores $\wedge$), poderá usar uma combinação de **Assoc** e **Comm** para rearranjar as variáveis em qualquer ordem que quiser, o que pode ser uma ferramenta muito poderosa para ajudar a moldar uma proposição em quase tudo que precisar.

Fazendo a Dist

Dist: $x \wedge (y \vee z)$ é equivalente a $(x \wedge y) \vee (x \wedge z)$

$x \vee (y \wedge z)$ é equivalente a $(x \vee z) \wedge (x \vee z)$

A **Dist** ainda tem outras duas formas que vale a pena conhecer, com as subproposições em parênteses na frente:

$(x \vee y) \wedge z$ é equivalente a $(x \wedge z) \vee (y \wedge z)$

$(x \wedge y) \vee z$ é equivalente a $(x \vee z) \wedge (y \vee z)$

LEMBRE-SE

A maioria dos professores acha essas outras duas formas da regra aceitáveis. Entretanto, alguns mais detalhistas podem querer que você use a **Comm** para transformar $(x \vee y) \wedge z$ em $z \wedge (x \vee y)$ antes de aplicar a **Dist**. Podem querer, alternativamente, que você use a **Comm** para transformar $(x \wedge y) \vee z$ em $z \wedge (x \wedge y)$ antes de aplicar a **Dist**.

Por exemplo, considere esta proposição:

$P \vee (Q \wedge (R \vee S))$

Você viu que podemos usar a **Dist** de duas maneiras aqui. Primeiro, considerando $(R \vee S)$ como uma só; observe que a proposição é assim:

$x \vee (y \wedge z)$

Agora, pode reescrever a proposição para que se pareça com isto:

$(P \vee Q) \wedge (P \vee (R \vee S))$

A vantagem aqui é que você pode usar a **Simp** para dividir a proposição em duas menores:

$P \vee Q$

$P \vee (R \vee S)$

A segunda opção é usar a **Dist** na subproposição $Q \wedge (R \vee S)$, que resulta em:

$$P \vee ((Q \wedge R) \vee (Q \wedge S))$$

Agora, você tem três subproposições – P, $(Q \wedge R)$ e $(Q \wedge S)$ – ligadas por proposições $\vee$, o que significa que pode usar a **Assoc** e a **Comm** para arranjá-las em qualquer ordem que quiser, desde que deixe intacto o que está entre parênteses.

Uma das grandes utilidades da **Dist** é que ela permite que você troque o operador principal de uma proposição de $\wedge$ para $\vee$. Ao trocá-lo dessa maneira, você poderá substituir uma conclusão não amigável, isto é, uma em que não pode usar a prova condicional, em uma amigável.

Por exemplo, considere esta proposição:

$$P \wedge (Q \vee R)$$

Na maioria dos casos, quando você enfrenta uma conclusão no formato $x \wedge y$, deve esquecer a prova condicional. Mas, neste caso, pode usar a **Dist** para transformá-la, assim:

$$(P \wedge Q) \vee (P \wedge R)$$

Agora, você tem a conclusão em uma forma amigável e pode usar a **Impl** para transformá-la em uma proposição $\rightarrow$:

$$\sim(P \wedge Q) \rightarrow (P \wedge R)$$

Paralelamente, outra forma não amigável é $\sim(x \rightarrow y)$. Conhecer essa forma pode fazer com que você abandone a prova condicional ao se deparar com a seguinte conclusão:

$$\sim(P \rightarrow \sim(Q \vee R))$$

Mas, felizmente, você pode tirar essa conclusão de sua forma negativa em dois passos. Primeiro, use a **Impl** para deixá-la assim:

$$\sim(\sim P \vee \sim(Q \vee R))$$

Depois, use a **DeM** :

$P \wedge (Q \vee R)$

Surpreendentemente, essa conclusão é a mesma com que comecei no exemplo de $P \wedge (Q \vee R)$, na página anterior, de forma que você pode usar a **Dist**, como fez lá, para transformá-la em uma forma amigável e, então, atacála com uma prova condicional.

Desmembre premissas longas

Como mencionei anteriormente, quebrar premissas longas pode ser difícil. Algumas vezes, porém, simplesmente não há como fazer isso. Observe, por exemplo, este argumento:

$(P \wedge Q) \vee (Q \to R), Q, \sim R$: P

1.	$(P \wedge Q) \vee (Q \to R)$	**P**
2.	Q	**P**
3.	$\sim R$	**P**

A chave aqui – quer seja pela prova direta ou indireta – é encontrar um jeito de quebrar essa premissa longa. Sugiro a prova direta.

Quando trabalhar com uma premissa longa, decida que forma de proposição ela é. Isso, em geral, ajuda a indicar o próximo passo a seguir.

A forma da primeira premissa é $x \vee y$. Quando você se depara com uma proposição $\vee$, primeiro deve tentar a **Impl**:

4.	$\sim(P \wedge Q) \to (Q \to R)$	1 **Impl**

Com isso, pode ver que se trata de uma proposição $x \to (y \to z)$, que lhe permite tentar a **Exp**:

5.	$(\sim(P \wedge Q) \wedge Q) \to R$	4 **Exp**

Agora, você está bem, pois isolou *R* como a segunda parte da proposição. Então, use a **MT**:

6. $\sim(\sim(P \wedge Q) \wedge Q)$ 3, 5 **MT**

Tenha o cuidado de não tentar cancelar um operador ~ com o outro. O primeiro se aplica à proposição inteira, e o segundo se aplica somente à subproposição $(P \wedge Q)$.

Agora, a forma da proposição é $\sim(x \wedge y)$, o que significa que você tem uma ótima oportunidade de usar a **DeM**:

7. $(P \wedge Q) \vee \sim Q$ 6 **DeM**

O resto dos passos se revelam sozinhos:

8. $P \wedge Q$ 2, 7 **SD**

9. P 8 **Simp**

Observe que prestar atenção na forma da proposição foi fundamental para destrinchar a premissa, camada por camada. Algumas vezes, quando estudantes se deparam com provas difíceis como essa, perguntam: “Mas e se eu não conseguir enxergar? E se não conseguir descobrir o próximo passo?”.

A boa notícia é que você quase sempre pode encontrar mais de uma maneira de fazer a prova. Por isso, mostrarei a seguir que mesmo quando começar de maneira diferente, geralmente conseguirá encontrar uma forma de fazê-la funcionar.

Veja a mesma prova que acabo de discutir, mas com um passo inicial diferente:

1.	$(P \wedge Q) \vee (Q \rightarrow R)$	**P**
2.	Q	**P**
3.	$\sim R$	**P**

Nesse caso, comece aplicando a **Impl**, não no operador principal, mas na segunda parte da proposição:

4.	$(P \wedge Q) \vee (\sim Q \vee R)$	1 **Impl**

Desta vez, você pode identificar a proposição como sendo da forma $x \vee (y \vee z)$, que significa que pode usar a **Assoc**:

5.	$((P \wedge Q) \vee \sim Q) \vee R$	4 **Assoc**

Surpresa! Você chegou, novamente, no ponto de usar a **SD**. Entretanto, desta vez pode usá-la duas vezes seguidas:

6.	$(P \wedge Q) \vee \sim Q$	3, 5 **SD**
7.	$P \wedge Q$	2, 6 **SD**

Novamente, a resposta surge:

8.	P	7 **Simp**

Faça uma suposição sagaz

No Capítulo 11, mostrei como usar a prova indireta, assumindo a negação da conclusão, para então refutá-la.

Mas, com a prova indireta, você não está limitado a usar a negação da conclusão. Na verdade, pode fazer *qualquer* suposição e tentar refutá-la. Se você conseguir, terá provado a *negação* da suposição, o que geralmente o ajuda a provar a conclusão.

Embora você possa fazer qualquer suposição, a estratégia aqui é escolher uma que o leve facilmente à contradição. Por exemplo, observe o argumento que provei ser válido no tópico anterior "Desmembre premissas longas":

$(P \wedge Q) \vee (Q \to R), Q, \sim R : P$

1.	$(P \wedge Q) \vee (Q \to R)$	**P**
2.	Q	**P**
3.	$\sim R$	**P**

Nessa prova, você vê uma maneira rápida de quebrar a primeira premissa e cria uma Premissa Presumida para ajudá-lo a fazer com que isso aconteça:

4.	$\sim(P \wedge Q)$	**PP**

Assim como acontece em todas as provas indiretas, você tem, agora, uma contradição. Mas, isso levará apenas algumas linhas:

5.	$Q \to R$	1, 4 **SD**
6.	R	2, 5 **MP**
7.	$R \wedge \sim R$	3, 6 **Conj**

Como sempre, o próximo passo é descartar a **PP**:

8.	$P \wedge Q$	4-7 **PI**

Agora, é muito simples concluir a prova:

9.	P	8 **Simp**

Capítulo 13

Um por Todos e Todos por Um

Neste Capítulo

- Entendendo a suficiência dos operadores da LS
- Criando sistemas de LS com menos de cinco operadores
- Avaliando o traço de Sheffer

No Capítulo 4, mostrei que a palavra *ou* tem dois significados diferentes em português: o *ou* inclusivo, que significa “ou esse, ou aquele ou ambos”, e o *ou* exclusivo, que quer dizer “ou esse ou aquele, mas não ambos”.

Falei também que na Lógica Sentencial (LS) o operador $\vee$ retira essa ambiguidade, pois sempre representa o *ou* inclusivo. Você pode ter pensado que isso era extremamente injusto e discriminatório. Na verdade, os mais radicais dentre vocês já devem ter cogitado em fazer um manifestação para a criação de um sexto operador na LS.

Mas, antes de ir para as ruas carregando cartazes de protesto e gritando palavras de ordem em defesa do *ou* exclusivo, leia este capítulo. Nas próximas páginas, discutirei como o *ou* exclusivo — assim como qualquer outro operador que você possa inventar — já está inserido nos cinco operadores da LS. Na realidade, mostrarei neste capítulo como esses cinco símbolos lhe permitem expressar *qualquer* função verdade possível que você possa imaginar.

Você também descobrirá como expressar qualquer função verdade possível com *menos* de cinco operadores de LS, e vai se surpreender ao perceber o quanto pode fazer com tão pouco.

Contentando-se com os Cinco Operadores da LS

Depois de trabalhar com os cinco operadores da LS por um tempo, você pode começar a se perguntar se mais alguns operadores tornariam a linguagem mais útil. Neste tópico, você verá que a resposta é um sonoro “não!”

Para provar esse ponto de vista, inventei um novo operador fictício — ? – apenas para discutir a questão. O operador ? funcionará como um *ou* exclusivo (veja o Capítulo 4), o que significa *ou...ou...mas não ambos.* Veja a tabela verdade para esse novo operador:

x	y	x ? y
V	V	F
V	F	V
F	V	V
F	F	F

Você pode usar esse recém-criado operador da mesma forma que usaria qualquer outro operador na LS. Por exemplo, combinando-o com os já conhecidos, para criar uma proposição como *(P ? Q) → P.*

Também pode usar a tabela verdade para descobrir sob quais interpretações o valor da proposição é **V** ou **F**:

P	*Q*	*(P*	?	*Q)*	→	*P*
V	V	V	F	V	V	V
V	F	V	V	F	V	V
F	V	F	V	V	F	F
F	F	F	F	F	V	F

Parece uma grande ideia. Então, por que essa moda não pegou? Simples, você não precisa de outro operador; pode obter o mesmo resultado usando os operadores padrão da LS:

x	*y*	*x* ? *y*	~	*(x*	↔	*y)*
V	V	F	F	V	V	V
V	F	V	V	V	F	F
F	V	V	V	F	F	V
F	F	F	F	F	V	F

Como você pode ver, as proposições *x* ? *y e ~(x ↔ y)* são semanticamente equivalentes (veja o Capítulo 6 para saber mais sobre ela). Quando isso acontece, uma pode ser substituída pela outra.

Por exemplo, você pode substituir a proposição *~(P ↔ Q) → P* pela *(P* ? *Q) → P*. Como a tabela a seguir confirma, elas são semanticamente equivalentes:

P	Q	(P	?	Q)	→	P	~	(P	↔	Q)	→	P
V	V	V	F	V	V	V	F	V	V	V	V	V
V	F	V	V	F	V	V	V	V	F	F	V	V
F	V	F	V	V	F	F	V	F	F	V	F	F
F	F	F	F	F	V	F	F	F	V	F	V	F

Assim você não precisa realmente do operador ? para representar o *ou* exclusivo. Os cinco operadores são suficientes para tudo o que precisar do ou exclusivo.

Na verdade, *qualquer* operador que você pudesse inventar poderia ser igualmente desnecessário. Isto é, os cinco operadores da LS são suficientes para representar qualquer proposição que você quiser expressar na LS sem recorrer a operadores adicionais.

Redução – Uma História Verídica

No tópico anterior, mostrei que o operador ? era desnecessário, demostrando como você pode representar a proposição *(x ? y)* como *~(x ↔ y).* Em outras palavras, mostrei que essas duas proposições são semanticamente equivalentes.

Quando duas proposições são semanticamente equivalentes, você pode substituir uma pela outra sempre que quiser. Esse poder de substituição é muito útil quando você faz as provas e traz também um resultado inesperado, como ilustra a fábula contada nos próximos tópicos.

Neste tópico, você usará seu conhecimento das regras de equivalência (veja o Capítulo 10) para saber como eliminar os operadores da LS e, ainda assim, expressar o que precisa na LS.

A tirania do poder

Suponha que você já esteja cansado do operador ↔, que está sempre atrasado para o trabalho e querendo dias de folga. Você está louco para encarnar o apresentador de “O Aprendiz” e dizer “Você está demitido!”, mas está preocupado que os outros operadores não sejam capazes de fazer o trabalho sozinhos. Aí, tem uma ideia.

Usando a regra de **Equiv** (veja o Capítulo 10), qualquer proposição do formato *x ↔ y* é equivalente a *(x ∧ y)* ∨ *(~x ∧ ~y)*. Devido a essa equivalência, você decide dar aos outros operadores novos nomes pomposos e trocar suas atribuições. De agora em diante, ao invés de usar o operador ↔, usará as combinações de outras variáveis e operadores.

Por exemplo, ao invés de escrever

P ∧ (Q ↔ R)

escreverá

$P \wedge ((Q \wedge R) \vee (\sim Q \vee \sim R))$

Da mesma forma que, ao invés de escrever

$(P \rightarrow \sim Q) \leftrightarrow (R \wedge S)$

escreverá

$((P \rightarrow \sim Q) \wedge (R \wedge S)) \vee (\sim(P \rightarrow \sim Q) \wedge \sim(R \wedge S))$

É um pouco estranho, mas funciona. Na verdade, a grande descoberta aqui é que você será capaz de fazer tudo o que fazia com cinco operadores usando somente quatro. Quando digo tudo, quero dizer *tudo mesmo.*

Os ventos da discórdia

O topo é um lugar solitário, mesmo para os operadores lógicos. Na primeira hora da manhã de segunda-feira, o operador → invade sua sala, sem bater. Ele não está nada satisfeito com a nova arrumação e, depois de muita gritaria, lhe dá um ultimato: “Ou você recontrata o operador ↔, ou me demite!”.

É lógico que você não gosta nada desse tipo de conversa, de forma que expulsa o operador → para fora do prédio, escoltado por um par de parênteses robustos (aqui é onde a história assume oficialmente seu lado fantasioso). Depois que a situação se acalma, você percebe que terá mais uma vaga para preencher.

Mas, de novo com a regra de **Impl** (veja o Capítulo 10), você sempre pode substituir qualquer proposição do formato $x \rightarrow y$ pela sua equivalente, $\sim x \vee y$. Isso significa, por exemplo, que pode escrever a proposição

$(((P \rightarrow Q) \wedge R) \rightarrow S)$

como

$(\sim((\sim P \vee Q) \wedge R) \vee S)$

Mais uma vez o sistema todo está funcionando. Você pode fazer tudo que fazia com cinco operadores usando apenas três (~, ^ e v).

Esses três operadores são suficientes para expressar todas as proposições da LS, o que é ainda mais importante na Álgebra Booleana, a primeira formalização rigorosa da Lógica. Veja o Capítulo 14 para saber mais sobre a Álgebra Booleana.

Entre a cruz e a espada: O dilema

Quando as coisas parecem estar voltando ao normal na fábula do operador, o ^ e o v solicitam uma reunião. Eles estão trabalhando há muitas horas, sabem que você precisa deles e, por isso, querem gordos aumentos de salário.

Você lhes oferece charutos e uísque, e diz que o pedido deles será sua prioridade máxima na próxima reunião de diretoria. Depois de terem saído de sua sala, você bola um plano para se livrar de um deles, para que possa dar um aumento de cinquenta por cento ao outro e ainda obter algum lucro no acordo.

Usando as Leis de DeMorgan (a **DeM** mostrada no Capítulo 10), você percebe que pode substituir qualquer proposição no formato $x \wedge y$ por sua equivalente $\sim(\sim x \vee \sim y)$. Mas também pode substituir qualquer proposição no formato $x \vee y$ por sua equivalente $\sim(\sim x \wedge \sim y)$.

Pela primeira vez, você hesita. Chega até a considerar a possibilidade de despedir os dois e recontratar o operador → (já que poderá substituir $x \wedge y$ por $\sim(x \rightarrow \sim y)$ e $x \vee y$ por $\sim x \vee y$).

De qualquer modo, você agora tem muito espaço para negociação. As três combinações a seguir permitem que você faça o trabalho de todos os cinco operadores usando apenas dois.

~ e ^

~ e v

~ e →

O traço do gênio (de Sheffer)

Você não previa, mas seu operador mais fiel, o “~”, lhe deu aviso prévio. Desta vez não é uma questão de dinheiro nem de horas extras, nem mesmo do clima no escritório. Na verdade, ele não quer nada além de aposentadoria antecipada, com um bom acordo financeiro.

Com essa notícia, é provável que você tenha que fechar as portas. Mesmo que todos os seus operadores voltem ao trabalho, não é possível negar uma proposição sem o operador ~.

Quando tudo parece perdido, um visitante surpresa aparece: o operador |. É o *nand* (abreviação de *não* e *e*, em inglês), também chamado de *traço de Sheffer*, em homenagem ao seu inventor, Henry Sheffer. A proposição *x* | *y* é semanticamente equivalente *a ~ (x ^ y).*

x	*y*	*x* | *y*	~	(*x*	&	*y*)
V	V	F	F	V	V	V
V	F	V	V	V	F	F
F	V	V	V	F	F	V
F	F	V	V	F	F	F

Contratar esse novo ajudante lhe trará algumas vantagens. Por exemplo, com ele você pode expressar uma negação, tal como ~*x*, usando *x* | *x*:

x	~x	x | x

V	F	F
F	V	V

Pode ainda expressar proposições $\wedge$, usando $(x \mid y) \mid (x \mid y)$, e proposições $\vee$, tais como $x \vee y$, usando $(x \mid x) \mid (y \mid y)$.

Na verdade, o traço de Sheffer lhe permite expressar o mesmo que os cinco operadores da LS usando apenas um operador. Por exemplo, considere a expressão:

$$X \rightarrow (Y \leftrightarrow R)$$

Comece por retirar o operador $\leftrightarrow$

$$X \rightarrow ((Y \wedge R) \vee (\sim Y \vee \sim R))$$

Depois remova o operador $\rightarrow$

$$\sim X \vee ((Y \wedge R) \vee (\sim Y \vee \sim R))$$

Agora cuide dos operadores $\wedge$ e $\vee$:

$$\sim X \vee (((Y \mid R) \mid (Y \mid R)) \vee (\sim Y \vee \sim R))$$

$$\sim X \vee (((Y \mid R) \mid (Y \mid R)) \vee (\sim Y \mid \sim Y) \mid (\sim R \mid \sim R))$$

$$\sim X \vee ((((Y \mid R) \mid (Y \mid R)) \mid ((Y \mid R) \mid (Y \mid R))) \mid (((\sim Y \mid \sim Y) \mid (\sim R \mid \sim R)) \mid ((\sim Y \mid \sim Y) \mid (\sim R \mid \sim R))))$$

$$(\sim X \mid \sim X) \mid (((((Y \mid R) \mid (Y \mid R)) \mid ((Y \mid R) \mid (Y \mid R))) \mid (((\sim Y \mid \sim Y) \mid (\sim R \mid \sim R)) \mid ((\sim Y \mid \sim Y) \mid (\sim R \mid \sim R)))) \mid ((((Y \mid R) \mid (Y \mid R)) \mid ((Y \mid R) \mid (Y \mid R))) \mid (((\sim Y \mid \sim Y) \mid (\sim R \mid \sim R)) \mid ((\sim Y \mid \sim Y) \mid (\sim R \mid \sim R)))))$$

Finalmente cuide dos operadores ~:

$$((X \mid X) \mid (X \mid X)) \mid (((((Y \mid R) \mid (Y \mid R)) \mid ((Y \mid R) \mid (Y \mid R))) \mid ((((Y \mid Y) \mid (Y \mid Y)) \mid ((R \mid R) \mid (R \mid R))) \mid (((Y \mid Y) \mid (Y \mid Y)) \mid ((R \mid R) \mid (R$$

| R))))) | ((((Y | R) | (Y | R)) | ((Y | R) | (Y | R))) | ((((Y | Y) | (Y | Y)) | ((R | R) | (R | R))) | (((Y | Y) | (Y | Y)) | ((R | R) | (R | R))))))

Tudo bem, é um pouco entediante, mas pode ser feito. Então, depois de pensar muito, você anuncia “Está contratado!”. As cortinas se fecham e todos aplaudem.

A moral da história

É óbvio que, na realidade, os cinco operadores não estão sob ameaça real de serem demitidos, e dificilmente conseguirão se aposentar tão cedo, o que é muito bom. Mesmo que o operador | seja logicamente capaz de lidar com todas as questões envolvendo a LS, o exemplo do tópico anterior mostra como tudo pode se complicar sem eles.

Fazendo uma analogia, se você é um gênio da computação, sabe que tudo o que faz no computador é traduzido para 1 e 0. Porém, o teclado do seu computador contém muito mais do que essas duas teclas.

Essa redundância pode ser logicamente desnecessária, mas torna o uso do computador muito mais simples (assim como os cinco operadores da LS facilitam a Lógica). Você não precisa usar a linguagem do computador, basta usar a sua.

De modo parecido, os cinco operadores da LS criam um paralelo entre as palavras *e, ou, não* e assim por diante, para tornar mais fácil para você pensar no significado do que está fazendo, e não nas regras envolvidas no processo.

Capítulo 14

Forma Sintática e Conteúdo Semântico

Neste Capítulo

- Entendendo sintaxe e semântica
- Descobrindo as fórmulas bem formadas (FPFs)
- Analisando a conexão simbólica entre a Álgebra Booleana e a LS

Steve Martin disse certa vez que você jamais ouviria pessoas dizendo: "Passe-me aquele piano."

A graça dessa proposição está na ideia de que, sob um ponto de vista, ela é totalmente normal e, sob outro, é completamente absurda. Este capítulo é sobre essas duas perspectivas, que incluem o seguinte:

- ✓ **Sintaxe:** é o nível da gramática em que a frase de Steve Martin parece normal, afinal de contas, "Passe-me aquele piano." é uma frase gramatical, desde a letra maiúscula do início até o ponto final. *Sintaxe* diz respeito à forma ou estrutura interna da linguagem, ou seja, as regras que tornam a linguagem possível.
- ✓ **Semântica:** é o nível do significado, nível em que a frase de Steve Martin se revela um tanto ridícula. Em outras palavras, você pode passar uma caneta, uma carteira ou até mesmo um gambá, mas não um piano. *Semântica* diz respeito à função, ou uso externo, da linguagem. Aqui, tudo diz respeito ao significado que torna a linguagem proveitosa.

Tanto para a descrição de uma linguagem natural, como o português, quanto para uma inventada, como a Lógica Sentencial (LS), a sintaxe e a semântica têm um importante papel. Neste capítulo, você entenderá as distinções principais entre as duas na LS. Discutirei as regras para as fórmulas bem formadas na LS, que explicam como distinguir uma proposição da LS de uma mera cadeia de símbolos. Finalmente, você conhecerá a Lógica Booleana, que é um sistema mais antigo da Lógica.

FBF – Você É Contra ou a Favor?

Tenho uma pergunta simples para você. Qual dos seguintes itens é uma proposição na LS?

> A) $(P \vee Q) \to {\sim}R$
>
> B) $\vee\ R\ Q \to\ )\ {\sim}(\ P$

Se escolheu *A*, acertou. Agora, uma pergunta um pouco mais difícil: como você sabia?

Você pode estar tentado a responder “dãa!”. Entretanto, vale ressaltar que *A* e *B* contêm os mesmos símbolos. Assim como os mesmos números de espaços entre os símbolos (quatro, mas quem está contando?). Certo, então *B* parece com *A* depois que saiu do liquidificador. Mas, esse é mesmo um bom motivo para rejeitá-la como uma proposição? Na verdade, é uma excelente razão.

A LS, como qualquer outra linguagem, é mais do que apenas uma porção de símbolos colocados juntos em qualquer posição. A ordem dos símbolos é mais uma das coisas que permitem que ela funcione. Se isso não fosse importante, você poderia escrever em português, assim: cachorro comida então Cristo Redentor cobra vinho cálice Barbra Streisand copo templo scao domu socma !faad-ahimdnr, ; j?jj;ag u, R.

Certo, a última “frase” não faz o menor sentido. Isso deve lembrá-lo de que a língua portuguesa também tem regras para combinar seus símbolos. Essas regras são comumente chamadas de gramática e, sem que você perceba, seu cérebro está repassando essas regras com grande velocidade e precisão, mesmo enquanto você lê este livro.

Gramática, também conhecida como sintaxe, é simplesmente um conjunto de regras para ordenar pequenos pedaços da linguagem. Na linguagem escrita, esses pedaços são chamados de palavras e pontuação. Na LS, são chamados de constantes, operadores e parênteses.

Neste tópico, você aprenderá como distinguir uma proposição na LS de uma habilidosa impostora. Aprenderá as regras específicas para a construção de um conjunto de LS, para descobrir se é uma proposição.

Entendendo as FBFs (e suas regrinhas básicas)

Qualquer série aleatória de símbolos da LS é chamada de *cadeia.* Algumas delas são proposições, outras, não. Essa decisão é uma questão *sintática*, isto é, que recai sobre a forma da cadeia de símbolos.

Quando você fala de sintaxe, é comum substituir a palavra *proposição* por *fórmula bem formada.* Observe que a palavra *forma* aparece duas vezes aqui, isso ocorre para fixar o entendimento de que a forma (em outras palavras, a sintaxe) de uma cadeia está em questão. A expressão *fórmula bem formada* é abreviada como FBF.

LEMBRE-SE

Toda cadeia de símbolos da LS pode ser ou não uma FBF.

Não se engane: na LS, as palavras *proposição* e *FBF* têm o mesmo significado. Mas, quando você estiver falando de cadeias em uma festa ou sarau de Lógica, é melhor usar *FBF* no lugar de *proposição*, e é isso que farei aqui.

Intuitivamente, você sabe que $(P \vee Q) \rightarrow \sim R$ é uma FBF, mas que $\vee RQ \rightarrow)\sim(P$ não é. Nesse caso, sua intuição está correta, mas o que exatamente a embasa? Aqui, mostrarei três regras simples para transformar essa intuição em conhecimento.

LEMBRE-SE

Com essas três regras simples, você pode construir qualquer proposição da LS vista neste livro, e qualquer outra que verá no futuro. Mais importante ainda, elas evitarão a construção de cadeias que não são uma FBF. Veja as três regras, sem mais delongas:

> ✓ **Regra 1:** qualquer constante (*P, Q R* e assim por diante) é uma FBF.
>
> ✓ **Regra 2:** se qualquer cadeia *x* é uma FBF, então ~*x* também é.
>
> ✓ **Regra 3:** se quaisquer duas cadeias *x* e *y* são FBF, então *(x ∧ y), (x* ∨ *y)*, *(x → y)* e *(x ↔ y)* são todas FBF.

Experimente agora: como construir *(P* ∨ *Q) → ~R?*

Pela Regra 1, *P, Q*, e *R* são todas FBFs:

P	*Q*	*R* R 1

Assim, pela Regra 2, a cadeia ~*R* também é uma FBF:

P	*Q*	*R* R 2

A Regra 3 diz que a cadeia *(P* ∨ *Q)* é uma FBF:

(P ∨ Q)	*~R*	R 3

Assim, aplicando a Regra 3, novamente, a cadeia *((P* ∨ *Q) → ~R)* é uma FBF.

((P ∨ *Q) → ~R)*	Regra 3

Flexibilizando as regras

Tecnicamente falando, toda FBF na LS deveria começar e terminar com parênteses. Mas, na prática, isso raramente acontece. Como você pode ver ao longo deste livro, as proposições raramente começam e terminam com parênteses.

LEMBRE-SE

Retirar os parênteses externos de uma FBF resulta naquilo que alguns lógicos chamam de versão *informal* da FBF.

Como você pode ver no exemplo do tópico anterior, as regras mostram que uma cadeia $((P \vee Q) \rightarrow \sim R)$ é uma FBF. Mas, ela é a mesma coisa que $(P \vee Q) \rightarrow \sim R$? A rigor, não. Então, a Regra 4 — também conhecida como *regras de flexibilização* — cuida disso.

Por convenção, com essa regra, você pode remover os parênteses externos de uma FBF para criar um versão mais informal dela. Tecnicamente, porém, essa versão mais informal não é uma FBF e a Regra 4 não é uma regra. Ao invés disso, é uma convenção que torna a FBF mais fácil de ser lida.

CUIDADO!

Você *não* pode usar a versão informal para construir uma nova FBF.

Separando o que é FBF do que não é

Lembre-se de que a finalidade dessas regras para a construção de proposições da LS não é somente permitir a construção de FBFs, mas, também, evitar a construção de uma cadeia de símbolos que se assemelhe a uma FBF (mas não seja). Observe a confusa cadeia a seguir:

$$\vee RQ \rightarrow)\ \sim(\ P$$

Você não tem muita esperança ao tentar construir essa coisa maluca. Mas, veja algo que se parece com uma FBF:

$$(P \vee \sim Q) \rightarrow (R \vee S) \wedge \sim T$$

É possível construir essa proposição usando as regras? Veja um caminho :

P	Q	R	S	T	Regra 1
P	$\sim Q$	R	S	$\sim T$	Regra 2
$(P \vee \sim Q)$ $(R \vee S)$ $\sim T$					Regra 3

Neste ponto, você tem todos os parênteses que queria adicionar, mas ainda estão faltando dois operadores. Mesmo se você fosse usar a versão informal da proposição, não poderia inserir dois operadores na proposição sem adicionar um novo conjunto de parênteses, o que é exatamente o ponto. A cadeia

$$(P \vee \sim Q) \rightarrow (R \vee S) \wedge \sim T$$

não é uma FBF. O operador principal poderia ser tanto o → quanto o ^. Essa ambiguidade é inaceitável, pois como visto no Capítulo 4, uma proposição da LS só pode ter um operador principal.

Decidir se uma cadeia é uma FBF, de certa maneira, é a mesma coisa que decidir se um navio está apto para navegar. Antes de colocar um barco na água, embarcar e sair remando por aí, é bom saber se há alguma rachadura ou furos no casco. Se achar algo, é melhor você consertar (se possível) *antes* de partir.

O que serve para os barcos na água também serve para as cadeias na LS. Se você pensar, equivocadamente, que uma proposição é uma fórmula bem formada (navegável) e tentar usá-la em uma tabela verdade ou em uma prova (navegar), vai ter problemas (afundar como uma pedra).

Comparando a LS e a Álgebra Booleana

No Capítulo 13, mostrei como você pode se virar na LS usando apenas três operadores (~, ∧ e ∨) e, ainda assim, conseguir fazer tudo aquilo que fazia com cinco.

A versão mais antiga da Lógica Formal, isto é, a Lógica com símbolos ao invés de palavras da língua portuguesa, aproveitou-se do fato de que você pode se virar com apenas três operadores. Como visto no Capítulo 2, a Álgebra Booleana, inventada por George Boole, foi a primeira tentativa de transformar a Lógica Filosófica em um sistema matemático rigoroso. De fato, essa forma de Lógica era mais relacionada à matemática que você aprende na escola do que à LS.

Acredite ou não, a Álgebra Booleana é, na verdade, muito mais fácil do que a Álgebra que você aprendeu (ou não) na escola. Primeiro, porque trabalha com apenas dois números: 1 e 0. Você também só precisa se preocupar com adição e multiplicação. Neste tópico, mostrarei as semelhanças entre a LS e a Álgebra Booleana.

Interpretando os sinais

A Álgebra Booleana é, na verdade, apenas uma versão da LS que usa símbolos diferentes, de forma que, neste tópico, começarei com a Álgebra Booleana, fazendo apenas algumas modificações na LS.

Na verdade, substituirei somente cinco símbolos, como ilustra a Tabela 14-1.

Tabela 14-1 Relacionando os Símbolos da Álgebra Booleana e da LS

Símbolo da LS	***Símbolo da Álgebra Booleana***
V	1
F	0
~	–
∧	X

v	+

LEMBRE-SE

As constantes na Álgebra Booleana são utilizadas da mesma forma que na LS, o que significa que quando você usar os novos símbolos, poderá elaborar as mesmas tabelas verdade (veja o Capítulo 6). Por exemplo, pode construir uma tabela verdade relacionando o operador ∧ da LS com a multiplicação booleana:

P	*P*	*P* ∧ *Q*
V	**V**	**V**
V	**F**	**F**
F	**V**	**F**
F	**F**	**F**

P	*Q*	*P* × *Q*
1	**1**	**1**
1	**0**	**0**
0	**1**	**0**
0	**0**	**0**

Lembre-se de que o operador ^ da LS corresponde à multiplicação (3) na Álgebra Booleana, pois como ^ significa *e*, você pode associá-lo de maneira equivocada à adição.

Observe que a multiplicação booleana, simbolizada pelo sinal de vezes (3), é exatamente a mesma multiplicação que a normal, mas usa apenas 1 e 0: 1 3 1 = 1, e qualquer coisa multiplicada por 0 é igual a 0.

Do mesmo modo, você pode construir a tabela verdade relacionando o operador ∨ na LS com a adição booleana:

P	*P*	*P* ∨ *Q*
V	**V**	**V**
V	**F**	**V**
F	**V**	**V**
F	**F**	**F**

P	*Q*	*P* + *Q*
1	**1**	**1**
1	**0**	**1**
0	**1**	**1**

0	**0**	**0**

Nesse caso, a adição booleana funciona do mesmo modo que a normal, com apenas uma exceção: na booleana, 1 + 1 = 1, porque ele é o número mais alto do sistema.

Finalmente, você pode construir a tabela verdade relacionando o operador ~ e a negação booleana:

P	*~P*
V	**F**
F	**V**

P	*−P*
1	**0**
0	**1**

Embora pareça estranho que na Álgebra Booleana -1 = 0 e -0 = 1, lembre-se de que 1 e 0, aqui, são semelhantes a **V** e **F** na LS. Assim, não é tão estranho quando você pensa que **~V** = **F e ~F** = **V**.

Usar o sinal de igual (=) com os outros símbolos matemáticos é bastante natural. Igualdade, na Álgebra Booleana, significa a mesma coisa que o valor verdade na LS. Você pode pensar, ainda, como o sinal de igualdade significando que duas proposições são semanticamente equivalentes, o que quer dizer que elas têm o mesmo valor verdade, não importando o valor de cada uma de suas variáveis (veja o Capítulo 6 para saber mais sobre equivalência semântica).

A Tabela 14-2 demonstra as relações entre a atribuição de valores verdade na LS e a igualdade na Álgebra Booleana.

Tabela 14-2 Usando o Sinal de Igualdade (=) na Álgebra Booleana

Valor Verdade na LS ou Equivalência	***Semântica Igualdade Booleana***
"O valor verdade de P é **V**."	$P = 1$
"O valor verdade de Q é **F**."	$Q = 0$
"O valor verdade de $P \vee \sim P$ é **V**."	$P + - P = 1$
"$P \wedge Q$ é semanticamente equivalente a $Q \wedge P$"	$P \times Q = Q \times P$
"$\sim(P \vee Q)$ é semanticamente equivalente a $\sim P \wedge \sim Q$"	$\sim(P + Q) = - P \times - Q$

Fazendo as contas

Na LS, você evita escrever cadeias que misturem constantes com os valores **V** e **F**. Na Álgebra Booleana, isso não é necessário. Na verdade, você pode descobrir muito sobre a Lógica misturando esses diferentes tipos de símbolos. Por exemplo,

$$P \times 0 = 0$$

faz com que você recorde que qualquer proposição $\wedge$ – lembre que o sinal de multiplicação substitui o operador $\wedge$ – que contenha uma subproposição falsa (0) é falsa, não importa como seja o restante dela. Da mesma forma, a equação

$$P + 0 = P$$

diz que quando uma proposição $\vee$ contém uma subproposição falsa, seu valor verdade depende daquele da outra subproposição. E a equação

$P \times 1 = P$

diz que quando uma proposição ∧ contém uma subproposição verdadeira (1), seu valor verdade depende do valor da outra subproposição.

Do mesmo modo, a equação

$P + 1 = 1$

diz que quando uma proposição ∨ contém uma subproposição verdadeira, a proposição também é verdadeira.

Entendendo os semianéis

Tanto a Álgebra Booleana quanto a Aritmética, nos números inteiros não negativos (0, 1, 2, 3 e assim por diante), são *semianéis*, o que significa que compartilham um conjunto de propriedades em comum.

Por exemplo, observe que as primeiras três equações do tópico anterior são também verdadeiras na aritmética convencional. Elas funcionam simultaneamente em dois níveis – como as expressões de verdades lógicas e de verdades aritméticas.

No Capítulo 10, mostro como as propiedades comutativa, associativa e distributiva transitam da Aritimética para a Lógica, na LS. A Tabela 14-3 apresenta uma pequena lista de propiedades importantes na Álgebra Booleana.

Tabela 14-3 Propriedades Comuns à Álgebra Booleana e à Aritmética (e Todos os Outros Semianéis)

Propriedade	***Adição***	***Multiplicação***
Elemento de identidade	$P + 0 = P$	$P \times 1 = P$
Aniquiladora	n/a	$P \times 0 = 0$
Comutativa	$P + Q = Q + P$	$P \times Q = Q \times P$

Associativa	$(P + Q) + R = P + (Q + R)$	$(P \times Q) \times R = P \times (Q \times R)$
Distributiva	$P \times (Q + R) = (P \times Q) + (P \times R)$	n/a

Esse conjunto de propriedades é suficiente para classificar a Álgebra Booleana e a Aritmética convencional como exemplos de *semianéis*. Todo semianel tem as cinco propriedades elencadas na Tabela 14-3.

Explorando a sintaxe e a semântica na Álgebra Booleana

A Álgebra Booleana traz uma oportunidade interessante para enfocar algumas diferenças entre a sintaxe e a semântica discutidas anteriormente neste capítulo.

A LS e a Álgebra Booleana são bastante diferentes com relação à sintaxe, mas muito semelhantes em termos de semântica.

No nível da sintaxe, a Álgebra Booleana e a LS são tão diferentes quanto francês e alemão: entender um não significa entender o outro.

Mas depois que você conhece as regras sintáticas de ambos, pode ver que os dois sistemas podem ser usados para expressar tipos semelhantes de ideias. A análise de como os símbolos expressam as ideias é a verdadeira essência da semântica. Observe como uma proposição na Álgebra Booleana tem dois significados distintos, independentes entre si. Considere, por exemplo, a seguinte proposição:

$$1 \times 0 = 0$$

Em um nível, essa proposição expressa uma verdade aritmética. Em outras palavras, quando você multiplica 1 por 0, o resultado é 0. Mas, em outro nível, ela expressa uma verdade lógica, significando que, quando você relaciona uma proposição verdadeira com uma falsa, usando a palavra *e*, o resultado é uma proposição falsa. Dependendo do contexto semântico em que esteja trabalhando, uma equação na Álgebra

Booleana funciona tanto para expressar uma verdade matemática
quanto Lógica.

Parte IV:

Lògica Quantitativa (LQ)

Nesta parte...

Na Parte IV, você aprofundará seus conhecimentos de Lógica com a descoberta da Lógica Quantitativa ou LQ. Ela usa tudo o que você já conhece da LS, mas amplia seus horizontes, permitindo-lhe lidar com uma maior gama de problemas. Na verdade, ela é poderosa o bastante para lidar com tudo aquilo que as formas anteriores de Lógica (todos os 2.000 anos) lidaram.

No Capítulo 15, você conhecerá os fundamentos básicos da LQ, incluindo dois novos tipos de operadores de quantificação. No Capítulo 16, descobrirá como traduzir as proposições do português para a LQ. No Capítulo 17, mostrarei como escrever provas em LQ usando as habilidades adquiridas com a LS. No Capítulo 18, apresentarei as relações e as identidades, que são os instrumentos que tornam a LQ mais expressiva. Finalmente, no Capítulo 19, você descobrirá como usar as árvores lógicas na LQ e as surpreendentes verdades sobre as árvores infinitas (gostou do suspense?).

Capítulo 15

Expressando Quantidade com Qualidade: Apresentando a Lógica Quantitativa

Neste Capítulo

- Uma visão geral da Lógica Quantitativa
- Indo além da LS com os quantificadores universais e existenciais
- Distinguindo proposições de formas proposicionais

No Capítulo 3, mostrei que a Lógica nada mais é do que decidir sobre a validade ou invalidade de um argumento. Assim, se você já leu o Capítulo 3 e eu lhe disser para analisar o argumento a seguir, provavelmente saberá me dizer que é um argumento perfeitamente válido:

Premissas:

Todos os meus filhos são honestos.

Pelo menos um dos meus filhos é advogado.

Conclusão:

Ao menos um advogado honesto existe.

Esse argumento é realmente válido. Entretanto, o problema surge quando você tenta usar a Lógica Sentencial (LS) para demonstrar sua validade. Nenhuma das três proposições contém palavras familiares como *e, ou, se* ou *não* para que você possa usar os cinco operadores da LS para expressá-las.

Assim, o máximo que você pode fazer é expressar essas proposições como constantes da LS.

Considere *C* = Todos os meus filhos são honestos.

Considere *L* = Pelo menos um dos meus filhos é advogado.

Considere *H* = Ao menos um advogado honesto existe.

Depois que você expressa as proposições como constantes, pode criar o seguinte argumento:

C, L : *H*

É óbvio que alguma coisa importante se perdeu na tradução do português para a LS, de modo que não importa o método que você usar (tabela verdade, tabela rápida, árvore lógica ou prova), esse argumento na LS será inválido. Assim, o argumento original é inválido (o que não é verdade) ou a LS está errada (o que também não é verdade). Talvez o que falte aqui seja uma terceira alternativa.

Este capítulo, assim como os Capítulos 16 a 19, trata dessa terceira alternativa mágica. Aqui está a solução para o seu problema. Neste capítulo, você entenderá por que esse argumento é válido sem destruir completamente tudo o que aprendeu sobre a Lógica até agora.

Ao invés disso, mostrarei como ampliar a linguagem formal lógica da LS para um poderoso instrumento chamado Lógica Quantitativa. Essa nova linguagem traz dois novos símbolos que lhe permitem expressar conceitos lógicos que vão além do alcance da LS. Prepare-se, pois um novo mundo lógico o espera!

Uma Rápida Olhada na Lógica Quantitativa

A *Lógica Quantitativa* (LQ) usa quase a mesma estrutura da sentencial (LS). Você deve se lembrar de que a LS tem constantes e variáveis (veja o Capítulo 4). Uma constante representa uma proposição em uma linguagem natural (como o português), enquanto as variáveis a representam na LS. A LQ também possui constantes e variáveis, mas as utiliza de modo diferente, o que permite dividir as proposições em partes menores para obter maior precisão.

Em alguns livros, a Lógica Quantitativa também é chamada de *Predicativa* ou *de Primeira Ordem.*

Usando constantes individuais e de predicado

A LQ possui dois tipos de constantes (ao invés de apenas uma, como na LS):

- ✓ **Constantes Individuais:** Representam o *sujeito* de uma frase, com *letra minúscula* variando de *a* a *u.*
- ✓ **Constantes de Predicado:** Representam o *predicado* de uma frase, com *letra maiúscula* variando de *A* a *Z.*

Assim como na LS, a linguagem formal lógica da LQ lhe permite traduzir proposições do português para símbolos. Você deve se lembrar de que na LS, ao traduzir uma proposição do português para a Lógica Formal, o primeiro passo é definir algumas constantes. Por exemplo, se quiser traduzir a seguinte proposição:

David é feliz.

Primeiro, defina uma constante individual para representar o sujeito da frase:

Considere *d* = David

Depois, defina uma constante de predicado para representar o predicado da frase:

Considere *Hx* = *x* é feliz

Para escrever a proposição na LQ, apenas substitua o *x* pela constante individual:

Hd

Em português, o sujeito geralmente vem antes do predicado. Porém, na LQ a ordem quase sempre é invertida. Essa inversão pode ser confusa no começo, portanto, tenha isso sempre em mente.

Você já pode ver, no exemplo, uma maior flexibilidade trazida pela LQ. Com a LS você só seria capaz de atribuir uma variável para a proposição inteira, como segue:

Considere *D* = David é feliz. (TRADUÇÃO DA LS)

Veja outro exemplo de como traduzir uma proposição do português para a LQ:

O gato estava no tapete.

Primeiro, defina as constantes individual e de predicado:

Considere *c* = o gato

Considere *Mx* = *x* estava no tapete.

Agora, você pode traduzir a proposição original como

Mc

Observe que essa tradução funciona bem, não importa o tamanho do sujeito e do predicado. Você pode até mesmo traduzir a verborrágica sentença a seguir:

> Minha estrepitosa tia de Minas Gerais, com seus três lamuriosos poodles, está partindo para uma viagem de três meses por seis países da Europa e da África.

Nesse caso, defina suas constantes da seguinte forma:

> Considere *a* = minha estrepitosa tia de Minas Gerais, com seus três lamuriosos poodles
>
> Considere *Lx* = *x* está partindo para uma viagem de três meses por seis países da Europa e da África.

Então, você pode facilmente traduzir a proposição como:

> *La*

Somente use uma constante individual para representar um coisa só — nunca mais de uma.

Assim, quando analisa o exemplo anterior, o sujeito da sentença é a tia, os poodles são apenas parte de sua descrição. Se a sentença fosse assim:

> Minha estrepitosa tia de Minas Gerais *e* seus três lamuriosos poodles *estão* partindo para uma viagem de três meses por seis países da Europa e da África.

Na LQ você não pode usar uma constante para representar um grupo de três poodles. Ao invés disso, precisa definir três outras constantes individuais, uma para cada um dos três poodles. Por exemplo, suponha que os nomes dos poodles sejam Fifi, Kiki e Elvis. Assim, você pode usar as seguintes definições:

> Considere *f* = Fifi

Considere k = Kiki

Considere e = Elvis

Então, se os três estivessem indo viajar juntos, a proposição ficaria assim:

$La \wedge Lf \wedge Lk \wedge Le$

Como você pode ver, o operador $\wedge$ da LS também pode ser usado na LQ. No próximo tópico, verá como todos os cinco operadores entram no jogo.

LEMBRE-SE

A definição formal de constantes é um ponto técnico importante, mas é muito fácil, e provavelmente você pegará o jeito muito rápido. Assim, de agora em diante, deixe essa parte de lado, a menos que seja necessária para o entendimento.

Incorporando os operadores da LS

LEMBRE-SE

A LQ contém os cinco operadores da LS: ~, $\wedge$, $\vee$, $\rightarrow$ e $\leftrightarrow$. Você pode reduzir grande parte do trabalho usando essa nova formulação de proposições, especialmente quando adiciona os operadores da LS que já conhece. Assim, com a LQ você pode traduzir sentenças para símbolos em um nível mais profundo do que era possível com a LS.

Por exemplo, considere a proposição “Nadine é contadora e sua secretária está de férias”. Na LS, você pode apenas quebrar a frase grosseiramente em duas partes:

$N \wedge S$ (TRADUÇÃO DA LS)

Com a LQ, porém, você pode distinguir quatro partes diferentes da sentença e usá-las para representá-la, da seguinte forma:

$An \wedge Ms$

Da mesma forma, a proposição

Gabriela é de Escorpião ou de Aquário.

pode ser traduzida como

$Sg \vee Ag$

Aqui, a frase é entendida como “*Ou* Gabriela é de Escorpião *ou* Gabriela é de Aquário”.

De modo semelhante, a proposição

Se Henry é um cara legal, então eu sou a Rainha de Sabá.

pode ser traduzida como

$Nh \to Qi$

Finalmente a proposição

Cinco é ímpar, se e somente se, ele não for divisível por dois.

pode ser traduzida como

$Of \leftrightarrow \sim Df$

Nessa frase, a palavra *ele* é um pronome que substitui o *cinco*, desta forma, você pode utilizar a constante individual *f* nos dois casos.

Como você pode ver nesses exemplos, constantes com uma única letra da LS foram substituídas por termos de duas letras na LQ. Mas não tema, você ainda pode combinar esses termos obedecendo as mesmas regras usadas na LS.

Entendendo as variáveis individuais

Na LQ, a *variável individual* representa um sujeito *inespecífico* de uma frase, usando uma letra minúscula *v, w, x, y* ou *z.*

Você já viu as variáveis individuais usadas na definição formal das constantes de predicado. Por exemplo, na definição,

Considere $Hx = x$ é feliz

a variável individual é *x*. Como viu anteriormente, você pode substituí-la por uma constante individual para criar uma proposição da LQ. Por exemplo, quando, na LQ representa a frase “David é feliz” como

Hd

você substitui a variável *x* pela constante *d*.

Você deve ter notado que a LQ tem constantes individuais, variáveis individuais e constantes de predicado, mas não *variáveis de predicado.* Elas só entrarão em cena na *Lógica Predicativa de Segunda Ordem* (veja o Capítulo 21), que é a razão pela qual a LQ também é chamada de *Lógica Predicativa de primeira ordem*.

Para economizar espaço, a expressão *variável individual* foi abreviada para *variável* ao longo de toda a Parte IV.

Expressando Quantidade com Dois Novos Operadores

Até agora, você viu neste capítulo alguns exemplos de como a LQ representa proposições que *podem* ser expressadas na LS. Agora, verá neste tópico como a LQ se sai no território em que a LS *não* alcança.

Os principais veículos para essa nova expressividade são os dois novos símbolos: $\forall$ e $\exists$.

Entendendo o quantificador universal

O símbolo $\forall$ é chamado de *quantificador universal*. Ele é sempre usado com uma variável, como *x*, sendo que $\forall x$ é traduzido para o português como "para todo *x...*".

No início deste capítulo, você viu como é problemático traduzir a proposição "todos os meus filhos são honestos" para LS de maneira útil. Mas com o quantificador universal, você terá esse poder.

Veja um exemplo: primeiro você precisa ligar um quantificador universal a uma variável, construindo o problema com colchetes vazios, como demonstrado a seguir:

$$\forall x\ [\ ]$$

Depois de elaborado o problema, pode usar as regras que já conhece para traduzir o resto da proposição. Coloque o resultado dentro dos colchetes:

$$\forall x\ [Cx \rightarrow Hx]$$

Você lê essa proposição como "para todo *x*, se *x* é meu filho, então *x* é honesto". A princípio, pode parecer um pouco desajeitado, mas, depois que você se acostuma, as possibilidades são literalmente infinitas.

Você também pode usar o $\forall$ para traduzir palavras como *todo* e *cada.* Essas traduções serão vistas mais detalhadamente no Capítulo 16.

Expressando existência

O símbolo $\exists$ é chamado de *quantificador existencial*. Ele é sempre usado com uma variável, como o *x*, sendo que $\exists x$ é traduzido para português como "existe um *x* tal que...".

No início deste capítulo, você viu que a LS é limitada em sua capacidade de expressar a proposição "ao menos um dos meus filhos é advogado". Entretanto, o quantificador existencial torna essa expressão possível.

Assim como ocorre com o quantificador universal, primeiro ligue o quantificador existencial a uma variável:

$$\exists x\ [\ ]$$

Agora dentro dos colchetes, traduza o resto da proposição:

$$\exists x\ [Cx \wedge Lx]$$

Você pode ler essa proposição como "existe um *x* tal que *x* é meu filho e *x* é advogado".

De forma semelhante, você pode usar $\exists$ para traduzir a proposição "ao menos um advogado honesto existe".

Assim, veja seu primeiro argumento em LQ:

$$\forall x\ [Cx \rightarrow Hx],\ \exists x\ [Cx \wedge Lx] : \exists x\ [Hx \wedge Lx]$$

Como você pode ver, acontece muito mais coisa aqui do que na versão da LS. Isso é bom, pois a LQ transformou em símbolos exatamente o que é válido nesse argumento. Porém, antes de pensar em como provar a validade desse argumento, você precisa de um pouco mais de experiência.

Você também pode usar o ∃ para traduzir palavras como *algum(a)* e *há.* As traduções desse tipo serão vistas no Capítulo 16.

Criando um contexto com o domínio de discurso

Tendo em vista que a LQ pode lidar com uma maior variedade de circunstâncias do que a LS, um novo tipo de ambiguidade surge quando ocorre a tradução do português para a LQ.

Por exemplo, observe esta proposição:

> Todos estão vestindo terno.

Você pode traduzir essa proposição para a LQ da seguinte maneira:

$$\forall x\ [Sx]$$

A tradução literal é “qualquer que seja *x*, *x* está vestindo um terno”, o que é falso. Mas, se o contexto de que estiver falando for o de uma reunião de colegas de trabalho do sexo masculino, a proposição pode ser verdadeira.

Para evitar essa possibilidade de confusão, você precisa ser claro sobre o contexto da proposição. Por isso deve inserir as variáveis no contexto chamado de *domínio de discurso.*

O domínio de discurso fornece o contexto para as proposições que contêm as variáveis. Por exemplo, se ele é *pessoas*, então *x* pode substituir *eu, você* ou sua tia Henrieta, mas não Bart Simpson, seu celular ou o planeta Júpiter.

Na Matemática, o domínio de discurso é, de modo geral, subentendido. Por exemplo, quando você escreve a equação $x + 2 = 5$, pode assumir que a variável x substitui um número, de forma que, nesse caso, o domínio de discurso é sempre o conjunto de todos os números.

O modo como você traduz uma proposição do português para a LQ depende geralmente do domínio de discurso. Para ver por que isso ocorre, observe as duas traduções de um argumento usando dois domínios de discurso diferentes:

Premissas:

Todos os meus filhos são honestos.

Pelo menos um dos meus filhos é advogado.

Conclusão:

Ao menos um advogado honesto existe.

Para a primeira tradução, veja a denominação utilizada:

Domínio: Irrestrito

Um domínio irrestrito significa que x pode ser, literalmente, qualquer coisa: uma pulga, um unicórnio, a Declaração de Independência, a galáxia de Andrômeda, a palavra *manteiga* ou seu primeiro beijo (você pensou que eu estava brincando quando disse *tudo*, não?). Essa amplitude de alcance não é problema, pois as próprias proposições restringem o que pode ser *x*.

Com esse domínio de discurso, o argumento é traduzido para a LQ como

$\forall x\ [Cx \rightarrow Hx], \exists x\ [Cx \wedge Lx] : \exists x\ [Hx \wedge Lx]$

Você pode pensar que o domínio de discurso é apenas uma questão técnica que não afeta realmente o argumento, mas não é. Na verdade, quando escreve um argumento, você pode usar o domínio de discurso em seu próprio benefício, escolhendo, sabiamente, um que torne a tradução do português para a LQ mais fácil e mais concisa.

Para a segunda tradução do mesmo argumento, posso começar declarando um domínio de discurso diferente:

Domínio: Meus filhos

Agora, todas as variáveis estão com uma restrição importante, o que significa que você pode traduzir o argumento "todos os meus filhos são honestos" como

$\forall x\ [Hx]$

ao invés do argumento usado anteriormente:

$\forall x\ [Cx \rightarrow Hx]$

Você interpreta $\forall x[Hx]$ como "qualquer que seja *x*, *x* é honesto" ou, simplesmente, "todo *x* é honesto". Observe que a proposição por si só não menciona o fato de que *x* é um dos meus filhos (Cx na proposição anterior), pois o domínio de discurso cuida dessa restrição.

Da mesma forma, usando o mesmo domínio de discurso, você pode traduzir a proposição "pelo menos um dos meus filhos é advogado" da seguinte maneira:

$\exists x\ [Lx]$

A interpretação dessa proposição é “existe um *x* tal que *x* é advogado”. Novamente, a proposição por si só não menciona que *x* é um dos meus filhos, mas essa restrição é feita pelo domínio de discurso.

Finalmente, você pode traduzir “ao menos um advogado honesto existe” como

$$\exists x\ [Hx \wedge Lx]$$

Essa é a mesma tradução de quando o domínio era irrestrito, pois a proposição não contém qualquer menção aos meus filhos. Assim, dentro do domínio “meus filhos”, o mesmo argumento é traduzido como

$$\forall x\ [Hx], \exists x\ [Lx] : \exists x\ [Hx \wedge Lx]$$

Como você pode ver, esse argumento é um pouco menor e mais simples do que aquele sob domínio irrestrito.

Certifique-se de usar apenas *um* domínio de discurso para um único argumento ou para quaisquer proposições que esteja analisando em grupo. Trocar de domínio não é a mesma coisa que tentar somar maçãs e laranjas, e o resultado, geralmente, é um argumento misturado que não é adequado.

O domínio de discurso é importante do ponto de vista técnico, mas na maioria das vezes, você pode usar o irrestrito. Então, de agora em diante, a menos que eu expressamente indique o contrário, presuma que o domínio é irrestrito.

Distinguindo Proposições de Formas Proposicionais

No Capítulo 5, descrevi a diferença entre uma proposição, como $(P \wedge Q) \rightarrow R$ e uma forma proposicional, como $(x \wedge y) \rightarrow z$, na LS. De forma simplificada, uma proposição tem constantes, mas não variáveis, e uma forma proposicional tem variáveis, e não constantes.

Na LQ, porém, um única expressão pode ter tanto constantes quanto variáveis. Neste tópico, explico como as constantes e as variáveis são usadas e no que consiste uma proposição na LQ.

Determinando o escopo do quantificador

No Capítulo 5, mostrei como os parênteses limitam o escopo de um operador na LS. Considere, por exemplo, a seguinte proposição da LS:

$$P \wedge (Q \vee R)$$

Estando *fora* dos parênteses, o operador $\wedge$ tem, como escopo, a proposição inteira, o que significa que ele afeta todas as variáveis nela contidas. Por outro lado, o escopo do operador $\vee$ é limitado ao que está *dentro* dos parênteses, visto que ele também está lá dentro. Isto é, esse operador afeta somente as constantes *Q* e *R*, mas não a *P*.

LEMBRE-SE

Na LQ, o escopo dos quantificadores $\forall$ e $\exists$ é limitado de forma semelhante, mas usando colchetes, e não parênteses. Observe esta proposição:

$$Ax\ [Cx \rightarrow Nx] \wedge Rb$$

O escopo do quantificador $\forall$ é limitado ao que está dentro dos colchetes: $Cx \rightarrow Nx$. Em outras palavras, esse quantificador não afeta o operador $\wedge$ ou a subproposição *Rb.*

Descobrindo as variáveis ligadas e livres

Pensou que já sabia o bastante sobre variáveis, não é mesmo? Prepare-se, pois aqui vem mais uma empolgante lição. Quando você estiver sentado à sua mesa, ponderando uma expressão, lembre-se de que toda variável naquela expressão é considerada *ligada* ou *livre*. Veja o que isso significa:

- ✓ Uma *variável ligada* é aquela que está sendo quantificada.
- ✓ Uma *variável livre* é aquela que não está sendo quantificada.

Fácil, não é mesmo? Para entender inteiramente a diferença entre variáveis ligadas e livres, observe a seguinte expressão:

$$\exists x\ [Dx \rightarrow My] \wedge Lx$$

Você consegue adivinhar aquelas que são ligadas e as que são livres?

LEMBRE-SE

Quando uma variável está *fora* dos colchetes, ela está sempre livre. Isso porque o escopo do quantificar é sempre limitado ao que está *dentro* do conjunto de colchetes. Então, x em Lx é uma variável livre.

Entretanto, você deve lembrar que apenas porque uma variável está dentro de colchetes não quer dizer que seja ligada. No exemplo, a variável y também é livre, mesmo estando dentro dos colchetes. Isso porque esses colchetes definem o escopo de $\exists x$, que está operando sobre x e não y.

Agora você já consegue adivinhar que a variável x em Dx é ligada. Pode estar certo disso, porque ela aparece dentro dos colchetes e porque eles definem o escopo de $\exists x$, que quantifica x.

A diferença entre variáveis ligadas e livres ficará mais importante no Capítulo 17, quando você mergulhar nas provas na LQ. Por ora, certifique-se de compreender que uma variável ligada está sendo quantificada e uma livre não.

Sabendo a diferença entre proposições e formas proposicionais

Com seu recente conhecimento sobre variáveis ligadas e livres aprendido no tópico anterior, você será capaz de distinguir facilmente uma proposição de uma forma proposicional.

LEMBRE-SE

Quando está valorando uma expressão, você tem sempre certeza de que ela se enquadra em uma das seguintes categorias:

> ✓ **Proposições:** *Não* contêm variáveis livres.
>
> ✓ **Formas Proposicionais:** Contêm pelo menos *uma* variável livre.

Distinguir uma proposição de uma forma proposicional é muito fácil. Considere, por exemplo, a expressão a seguir:

$$\forall x\ [Cx \rightarrow Hx] \wedge Ok$$

Essa expressão é uma proposição porque não possui variáveis livres. Observe que ambas as variáveis x aparecem dentro de colchetes, no escopo do quantificador $\forall x$. Além do mais, k aparece fora dos colchetes, mas é uma constante e não uma variável.

Por outro lado, considere a expressão:

$$\forall x\ [Cx \rightarrow Hx] \wedge Ox$$

Essa expressão tem uma variável livre (o x em Ox), que está fora do colchete e, portanto, fora do escopo do quantificador $\forall x$. Em consequência, a expressão é uma forma proposicional e não uma proposição.

Definições formais na LQ usam as formas proposicionais que podem ser transformadas em proposições. Observe esta definição:

Considere Ix = x é italiano.

A expressão Ix é uma forma proposicional, pois a variável x está livre. Mas você pode transformá-la em proposição, substituindo a variável pela constante individual. Por exemplo, para traduzir a proposição:

Ana é italiana.

Você pode escrever:

Ia

Da mesma forma, ligar uma variável a uma forma proposicional também a transforma em proposição. Para traduzir a proposição

Alguém é italiano.

você pode escrever

$\exists x\ [Ix]$

A diferença entre proposição e forma proposicional se tornará importante no Capítulo 18, quando você começará a escrever as provas em LQ. Por ora, basta compreender que a proposição não tem variáveis livres.

Capítulo 16
Traduções na LQ

Neste Capítulo

- Reconhecendo as quatro formas básicas de proposições categóricas, tais como *todos(as), algum(a)(s), nem todos(as)* e *nenhum(a)*
- Usando os quantificadores ∀ e ∃ para traduzir cada uma das formas básicas
- Traduzindo proposições em português que começam com palavras diferentes de: *todos(as), algum(a)(s), nem todos(as)* ou *nenhum(a)*

No Capítulo 15, apresentei um pouco da Lógica Quantitativa a (LQ), com especial atenção aos dois quantificadores ∀ e ∃. Mostrei, ainda, como traduzir algumas proposições simples para a LQ.

Neste capítulo, mostro como traduzir as quatro formas mais básicas de *proposições categóricas* do português para a LQ. Elas tendem a começar com palavras tais como *todos(as), algum(a)(s), nem todos(as), nenhum(a)* — veja o Capítulo 2 para saber mais sobre as proposições categóricas.

Este capítulo incluirá a tradução de cada uma dessas quatro formas, usando os quantificadores ∀ ou ∃. Finalmente, mostrarei como reconhecer as proposições categóricas que não começam com uma das quatro palavras-chave.

Traduzindo as Quatro Formas Básicas de Proposições Categóricas

Neste tópico, mostro como traduzir as palavras *todos(as), algum(a)(s), nem todos(as)* e *nenhum(a)*. Você usará bastante essas formas enquanto aprender a traduzir do português para a LQ (se precisar, dê uma revisada no Capítulo 2 para ter certeza de que entendeu as formas básicas). Depois, você estudará a tradução das proposições iniciadas por *todos(as)* e *algum(a)(s)* para, então, passar para aquelas começadas com *nem todos(as)* e *nenhum(a)*, que ficarão mais fáceis de entender.

Depois que souber como traduzir proposições iniciadas com palavras *todos(as), algum(a)(s), nem todos(as)* e *nenhum(a)*, o trabalho mais pesado estará feito. Em geral, a abordagem mais fácil desses tipos de proposições é montá-las com quantificador e colchetes, como na Tabela 16-1:

Tabela 16-1 Traduções das Quatro Formas Básicas de Proposições Categóricas

Palavra em Português	***Quantificador da LQ***
Todos(as)	$\forall$ [ ]
Algum(a)(s)	$\exists$ [ ]
Nem todos(as)	$\sim\forall x$ [ ]
Nenhum(a)	$\sim\exists\ x$ [ ]

"Todos(as)" e "algum(a)(s)"

Traduzir proposições do português para a LQ não é difícil, mas você tem que ter cuidado com alguns pontos complicados com relação ao domínio restrito e irrestrito (para refrescar a memória sobre como usar o domínio de discurso, veja o Capítulo 15).

Neste tópico, mostro como traduzir para a LQ proposições iniciadas com as palavras *todos(as)* e *algum(a)(s).*

Usando um domínio restrito

Duas proposições em português podem ser idênticas, exceto pelas palavras *todos(as)* ou *alguns(mas).* Por essa razão, você pode pensar que suas traduções para a LQ seriam idênticas, exceto pela escolha do quantificador ($\forall$ ou $\exists$). Quando o domínio é restrito, isso é o que geralmente acontece.

Por exemplo, considere as duas seguintes proposições:

Todos os cavalos são marrons.

Alguns cavalos são marrons.

Usando um domínio de discurso restrito:

Domínio: Cavalos

Se você usar um domínio restrito, precisará definir somente uma constante de predicado para traduzir as duas proposições. Por exemplo:

Considere Bx = x é marrom

Agora você pode traduzir ambas as proposições assim:

$\forall x\ [Bx]$

$\exists x\ [Bx]$

Como pode ver, a única diferença entre as duas proposições de LQ é que você substituiu o quantificador $\forall$ da primeira proposição pelo $\exists$ da segunda. Assim, como era de se esperar, não aconteceu nada de muito complicado aqui.

No restante do capítulo enfocarei as traduções em domínios irrestritos.

Usando um domínio irrestrito

Restringir o domínio nem sempre é possível, mas agora vamos imaginar que o domínio de discurso seja irrestrito.

Domínio: Irrestrito

Agora, para traduzir as proposições do tópico anterior, você precisa de outra constante de predicado. Assim, veja o exemplo, incluindo a constante utilizada no tópico anterior:

Considere Bx = x é marrom

Considere Hx = x é um cavalo

Considere Fx = x pode voar.

Agora você pode traduzir a proposição “todos os cavalos são marrons”, como

$\forall x\ [Hx \rightarrow Bx]$

Essa proposição é traduzida literalmente como “qualquer que seja *x*, se *x* é um cavalo, então *x* é marrom”.

Mas, por outro lado, você traduziu a proposição “alguns cavalos são marrons”, como

$\exists x\ [Hx \wedge Bx]$

Essa proposição é traduzida literalmente como “existe um *x* tal que *x* é um cavalo e *x* é marrom”.

Observe que a diferença entre as duas proposições é mais do que apenas os dois quantificadores $\forall$ e $\exists$. Na primeira proposição, você usou o operador $\rightarrow$, enquanto que na segunda usou o $\wedge$.

Veja o que acontece, porém, se misturar esses operadores. Se tentar traduzir a primeira proposição como

$\forall x\ [Hx \wedge Bx]$ ERRADO!

Essa tradução está errada porque, na verdade, está dizendo “qualquer que seja *x*, *x* é tanto cavalo quanto marrom”. De um modo mais informal: “tudo é um cavalo marrom”. Como o domínio é irrestrito, tudo quer realmente dizer *tudo*, o que obviamente não é o que você quer dizer.

Como outro exemplo, se você tentar traduzir a proposição

Alguns cavalos podem voar.

como

$\exists x\ [Hx \rightarrow Fx]$ ERRADO!

De novo essa tradução está errada porque, na verdade, está dizendo: “existe um *x* tal que, *se x* é um cavalo, então *x* pode voar”. O problema aqui é um pouco mais sutil.

Se *x* é algo que não um cavalo — por exemplo, um sapato —, então, a primeira parte da proposição dentro dos colchetes é falsa, o que torna tudo dentro dos colchetes verdadeiro. Nesse caso, a proposição parece estar dizendo que a existência de um sapato significaria que cavalos podem voar. De novo algo deu errado com a tradução.

LEMBRE-SE

Quando você traduzir do português para a LS, use as seguintes regras gerais:

- ✓ Quando usar $\forall$, use uma proposição $\rightarrow$ dentro dos colchetes.
- ✓ Quando usar $\exists$, use uma proposição $\wedge$ dentro dos colchetes.

“Nem todos(as)” e “nenhum(a)”

Depois que você souber como traduzir as duas formas positivas de proposições categóricas, as formas negativas — *nem todos(as)* e *nenhum(a)* — serão fáceis. Como vimos no Capítulo 2, uma proposição com *nem todos* é apenas a forma contraditória de um proposição com *todos(as)*. Do mesmo modo, uma proposição *nenhum(a)* é a forma contraditória de *algum(a)(s)*.

Por exemplo, você já sabe como traduzir a proposição

Todos os cavalos são marrons.

como

$\forall x\ [Hx \rightarrow Bx]$

Assim, sua forma contraditória é

Nem todos os cavalos são marrons.

Essa proposição é traduzida facilmente assim:

$\sim\forall x\ [Hx \rightarrow Bx]$

Além do mais, já que *todos(as)* e *nem todos(as)* são proposições contraditórias, você sabe com exatidão que uma delas é verdadeira e a outra é falsa.

Da mesma forma, sabe como traduzir a proposição

Alguns cavalos podem voar.

como

$\exists x\ [Hx \wedge Fx]$

A forma contraditória dessa proposição é

Nenhum cavalo pode voar.

Como você deve ter adivinhado, essa proposição é traduzida como

$\sim\exists x\ [Hx \wedge Fx]$

Novamente, porque *algum(a)(s)* e *nenhum(a)* são contraditórias, uma delas é verdadeira e outra é falsa.

Descobrindo Traduções Alternativas de Formas Básicas

A LQ é muito versátil e oferece mais de uma maneira de traduzir cada uma das formas. Neste tópico, mostro como traduzir cada uma das quatro formas básicas de proposições categóricas usando os dois quantificadores $\forall$ e $\exists$.

Conhecer ambas as versões de cada tradução o ajudará a entender a conexão oculta entre os dois quantificadores $\forall$ e $\exists$.

Para os exemplos neste tópico usarei as seguintes definições:

Considere *Dx* = *x* é um cachorro

Considere *Px* = *x* é brincalhão

A Tabela 16-2 organiza a informação contida neste tópico sobre a tradução das quatro formas básicas do português para a LQ. Para cada uma delas listo duas proposições em português. A primeira é o modo mais simples de expressá-la em português e sua tradução mais direta na LQ. A segunda dá uma construção alternativa da mesma ideia e sua tradução em LQ.

Tabela 16-2 Traduções Alternativas das Quatro Formas Básicas de Proposições Categóricas

Traduções em Português	***Traduções em LQ***
Todos os cachorros são brincalhões. (Nenhum cachorro não é brincalhão.)	$\forall\ [Dx \rightarrow Px]$ $(\sim\exists\ [Dx \wedge \sim Px])$
Alguns cachorros são brincalhões. (Nem todos os cachorros não são brincalhões.)	$\exists\ [Dx \wedge \sim Px]$ $(\forall\ [Dx \rightarrow \sim Px])$
Nem todos os cachorros são brincalhões. (Alguns cachorros não são brincalhões.)	$\sim\forall\ [Dx \rightarrow Px]$ $(\exists\ [Dx \wedge \sim Px])$

Nenhum cachorro é brincalhão.	$\sim\exists\ [Dx \land Px]$
(Todos os cachorro não são brincalhões.)	$(\forall\ [Dx \rightarrow \sim Px])$

Traduzindo "todos(as)" usando o ∃

Faz todo sentido traduzir a proposição "todos os cachorros são brincalhões" como

$\forall x\ [Dx \rightarrow Px]$

Uma outra tradução possível para o símbolo ∀ é *todos.* Por isso é fácil lembrar que é ele que geralmente usamos quando a palavra *todos* aparece na proposição.

Mas quando você pensa um pouco mais, percebe que quando diz que todos os cachorros são brincalhões, está construindo uma proposição equivalente:

Nenhum cachorro *não* é brincalhão.

Em outras palavras, essa proposição é do tipo *nenhum*, com a palavra *não* acrescentada. De forma que uma tradução perfeitamente possível é:

$\sim\exists x\ [Dx \wedge \sim Px]$

A tradução literal aqui é: "não é verdade que existe um *x* tal que *x* é um cachorro e *x* não é brincalhão". Como as duas proposições significam a mesma coisa em português, ambas são semanticamente equivalentes na LQ (para saber mais sobre equivalência semântica, veja o Capítulo 6).

Traduzindo "algum(a)(s)" usando o ∀

Da mesma maneira que você pode traduzir *todos(as)* usando o ∃, pode usar o quantificador ∀ para traduzir a palavra *algum(a)(s).* Por exemplo, considere a proposição

Alguns cachorros são brincalhões.

A tradução direta dessa proposição para a LQ é

$\exists x\ [Dx \wedge Px]$

Nesse caso, observe que essa proposição significa a mesma coisa que

Nem todos os cachorros *não* são brincalhões.

Assim, você pode tratar essa proposição da mesma forma que uma do tipo *nem todos*, com a adição da palavra *não e* traduzindo para:

$\sim\forall x\ [Dx \to \sim Px]$

Nesse caso, a tradução literal é "não é verdade que qualquer que seja *x*, se *x* é um cachorro, então *x* não é brincalhão". Como as duas proposições em português significam a mesma coisa, são semanticamente equivalentes na LQ.

Traduzindo "nem todos(as)" usando o ∃

Se você quiser traduzir a seguinte proposição:

Nem todos os cachorros são brincalhões.

Basta lembrar que já sabe traduzir essa proposição como

$\sim\forall x\ [Dx \to Px]$

Mas existe um outro jeito de pensar nela: já que nem todos os cachorros são brincalhões, *existe pelo menos um* cachorro, em algum lugar, que *não* é brincalhão. Em consequência, você pode reescrever a proposição original como

Alguns cachorros não são brincalhões.

Para expressar essa ideia, você pode usar a seguinte tradução:

$\exists x\ [Dx \wedge \sim Px]$

A tradução mais exata para essa proposição é "existe um *x* tal que *x* é um cachorro e *x* não é brincalhão". Essas duas proposições significam a mesma coisa em português, assim como suas correspondentes na LQ.

Traduzindo "nenhum(a)" usando o ∀

Se você quiser traduzir esta proposição:

Nenhum cachorro é brincalhão.

A tradução mais simples é

$\sim\exists x\ [Dx \wedge Px]$

A tradução mais exata aqui é “não é verdade que existe um *x* tal que *x* é um cachorro e *x* é brincalhão”.

Como deve ter adivinhado, você pode pensar nessa tradução de uma outra maneira. Por exemplo, o fato de que nenhum cachorro é brincalhão significa o seguinte:

Todos os cachorros não são brincalhões.

Você pode expressar essa ideia em LQ, como

$\forall x\ [Dx \rightarrow \sim Px]$

Nesse caso, a tradução mais exata é: “para todo *x*, se *x* é um cachorro, então *x* não é brincalhão”.

Mais uma vez, porque as duas proposições em português significam a mesma coisa, suas traduções na LQ são semanticamente equivalentes.

Identificando Proposições Disfarçadas

Você já sabe como traduzir as quatro formas básicas de proposições categóricas do português para o LQ. Agora vai se deliciar ampliando esses conhecimentos neste tópico.

Aqui, lhe mostro como reconhecer as proposições que não começam exatamente com as palavras que já conhece. Quando terminar, você estará pronto para traduzir uma variedade muito maior de proposições do português para a LQ.

Reconhecendo proposições do tipo "todos(as)"

LEMBRE-SE

Mesmo que uma proposição não comece com a palavra *todos(as)*, pode significar algo tão semelhante que justifique o uso do quantificador ∀. Veja alguns exemplos, seguidos de suas traduções para LQ:

Qualquer pai arriscaria sua vida para salvar um filho.

$\forall x\ [Fx \rightarrow Rx]$

Cada homem nesta sala é solteiro e desejável.

$\forall x\ [Mx \rightarrow (Sx \wedge Ex)]$

Família que reza unida permanece unida; enquanto que toda família que viaja de férias junta, sempre se diverte muito.

$\forall x\ [Px \rightarrow Sx] \wedge \forall x\ [Vx \rightarrow Lx]$

LEMBRE-SE

No exemplo anterior, você pode quantificar a variável *x* duas vezes, pois está construindo uma proposição sobre *todas* as famílias.

Reconhecendo proposições do tipo "algum(a)(s)"

Você pode se deparar com certas proposições que são bem parecidas com aquelas começadas com *algum(a)(s)*, onde usará o quantificador $\exists$ para traduzi-las. Por exemplo, considere essas proposições e suas respectivas traduções:

> Ao menos um dos convidados desta festa cometeu o crime.
>
> $\exists x\ [Gx \wedge Cx]$
>
> Muitos adolescentes são rebeldes e teimosos.
>
> $\exists x\ [Tx \wedge (Rx \wedge Hx)]$
>
> Tanto existem encanadores corruptos quanto juízes honestos.
>
> $\exists x\ [Px \wedge Cx] \wedge \exists y\ [Jy \wedge Hy]$

Observe que, no último exemplo, você precisou de duas variáveis — *x* e *y.* Elas deixam claro que ao mesmo tempo em que os dois grupos existem (encanadores corruptos e juízes honestos), eles não se sobrepõem necessariamente.

Reconhecendo proposições do tipo “nem todos(as)”

Você pode traduzir facilmente algumas proposições que não incluem as palavras *nem todos(as)*, usando $\sim\forall$. Por exemplo, considere estas proposições que são consideradas do tipo *nem todos:*

> Nem todo músico é excêntrico.
>
> $\sim\forall x\ [Mx \rightarrow Fx]$
>
> Pouco menos de 100 por cento dos contadores são reservados e meticulosos.
>
> $\sim\forall x\ [Ax \rightarrow (Rx \wedge Tx)]$
>
> Todas as coisas que reluzem não são ouro.
>
> $\sim\forall x\ [Ix \rightarrow Ox]$

Muito embora esse último exemplo comece com a palavra *todas*, ela é uma proposição do tipo *nem todos*, que significa “nem todas as coisas que reluzem são

ouro”. Sendo assim, pense bem sobre o que a proposição está realmente dizendo!

Reconhecendo proposições do tipo “nenhum(a)”

Você provavelmente já deve ter adivinhado que algumas proposições que não começam com a palavra *nenhum(a)* são facilmente traduzidas usando o $\sim\exists$; e está certo! Considere estas proposições e suas traduções:

Nem uma única pessoa do júri votou para absolver o réu.

$\sim\exists x\ [Jx \wedge Vx]$

Não existem quaisquer doutores nem mesmo bacharéis em nossa família.

$\sim\exists x\ [Fx \wedge (Dx \vee Cx)]$

Ninguém que compra um restaurante ganha dinheiro sem trabalhar muito.

$\sim\exists x\ [Bx \wedge (Mx \wedge \sim Px)]$

Você pode pensar nessa última proposição como “*nenhuma* pessoa existe que compre um restaurante *e* ganhe dinheiro *e* não trabalhe muito”.

Capítulo 17

Provando Argumentos com LQ

Neste Capítulo

- Comparando as provas na LS e na LQ
- Aplicando a negação do quantificador (**NQ**) na LQ
- Usando as quatro regras de quantificação **IU**, **IE**, **GU** e **GE**

Boas notícias: se já conseguiu entender as provas na Lógica Sentencial (LS), conhece oitenta por cento do que é necessário para escrever provas na Lógica Quantificativa (LQ). Assim, primeiro precisa descobrir o quanto você já sabe. Se precisar refrescar a memória, revise os capítulos da Parte III, que contêm tudo o que precisa saber.

Neste capítulo, começarei mostrando como as provas na LQ são semelhantes às da LS. Uma dessas semelhanças é que ambas usam as oito regras de implicação e as dez de equivalência; mostrarei exatamente quando usar cada uma delas.

Depois que você se familiarizar com algumas provas simples na LQ, conhecerá um regra adicional, a Negação do Quantificador (**NQ**), que permite fazer alterações que afetam o quantificador ($\forall$ ou $\exists$). Felizmente, essa regra, que é exclusiva da LQ, é muito fácil de entender.

O restante do capítulo enfoca as quatro regras de quantificação: Instanciação Universal (**IU**), Instanciação Existencial (**IE**), Generalização Universal (**GU**) e Generalização Existencial (**GE**). As primeiras duas permitem retirar cada um dos tipos de quantificador de uma proposição da LQ. As demais permitem adicionar cada um dos quantificadores à uma proposição LQ.

Em grupo, as quatro regras de quantificação permitem usar as oito regras de implicação da LS em sua capacidade plena. Continue lendo para descobrir como.

Aplicando as Regras da LS na LQ

As 18 regras de inferência da LS representam grande parte dos oitenta por cento que você já sabe sobre as provas na LQ. Elas incluem as oito regras de implicação do Capítulo 9 e as dez regras de equivalência do Capítulo 10. Em muitos casos, você pode transferir essas regras para as provas da LQ, fazendo apenas alguns pequenos ajustes.

Este tópico mostra as muitas semelhanças entre as provas na LQ e na LS.

Comparando proposições semelhantes da LS e da LQ

Uma das principais diferenças entre a LS e a LQ é o modo como essas duas linguagens lidam com proposições simples. Por exemplo, para traduzir a proposição "Heitor está dormindo" para LS, você pode escrever:

H (TRADUÇÃO PARA LS)

Para traduzir a mesma proposição para LQ, pode escrever:

Sh (TRADUÇÃO PARA LQ)

Da mesma forma, para traduzir uma proposição mais complexa "Heitor está dormindo e Emília está acordada" para a LQ, pode escrever:

$H \wedge E$ (TRADUÇÃO PARA LS)

Para traduzir a mesma proposição para a LQ, pode escrever:

$Sh \wedge Ae$ (TRADUÇÃO PARA LQ)

Nenhuma dessas proposições em português contém qualquer palavra que requeira o uso do quantificador (∀

LEMBRE-SE

ou ∃). É por isso que você pode traduzir ambas tanto para a LS quanto para a LQ.

De fato, nesses casos existe apenas uma diferença entre as traduções para LS e LQ: as constantes. Por exemplo, na LS você traduz uma proposição simples em português para uma constante de uma única letra (por exemplo, *H* ou *E*), que pode, depois, substituir sozinha uma proposição da LS.

Na LQ, porém, pode traduzir a mesma proposição em português para uma combinação de duas letras, uma constante de predicado e uma constante individual (por exemplo, *Sh* ou *Ae*), que podem, em seguida, substituir uma proposição da LQ (veja o Capítulo 4 para saber mais sobre as constantes na LS e o Capítulo 15 para as constantes da LQ).

DICA

Para as proposições da LQ, a regra básica é lidar com as combinações de duas letras como blocos *indivisíveis* de proposições maiores. Já que você não vai dividi-las, só pode lidar com elas como unidades, da mesma forma que trata as constantes de uma letra na LS (veja o Capítulo 18 para aprender sobre identidades, a única exceção a essa regra).

Transferindo as oito regras de implicação da LS para a LQ

LEMBRE-SE

Tendo em vista que a LS e a LQ são muito parecidas, você pode aplicar facilmente as oito regras de implicação a qualquer proposição ou forma proposicional da LQ, do mesmo modo que faria na LS.

CUIDADO!

Como na LS, quando usa as oito regras de implicação na LQ, você pode aplicá-las apenas às proposições inteiras ou formas proposicionais, nunca a partes delas (veja o Capítulo 9 para rever o uso das regras de implicação).

Trabalhando com as proposições LQ sem quantificador

Aqui mostro como as oito regras de implicação são aplicadas nas proposições da LQ sem quantificador. Por exemplo, se quiser provar este argumento:

$Jn \rightarrow Bn, \sim Bn \vee Ep, Jn : Ep$

Comece, como sempre, listando as premissas:

1.	$Jn \rightarrow Bn$	**P**
2.	$\sim Bn \vee Ep$	**P**
3.	Jn	**P**

Agora, continue da mesma maneira que na prova da LS, mas trate as constantes de duas letras como porções únicas, da mesma forma que trataria as constantes de uma letra só e as variáveis na LQ:

4.	Bn	1, 3 **MP**
5.	Ep	2, 4 **SD**

Trabalhando com as proposições na LQ com quantificador

As oito regras de implicação da LS também funcionam nas proposições da LQ com quantificador. Como ressaltei no tópico anterior, certifique-se de aplicar as regras em proposições e formas proposicionais inteiras.

Por exemplo, para provar o seguinte argumento:

$\forall x\ [Nx] \wedge \exists y\ [Py] : (\forall x\ [Nx] \vee \forall y\ [Py]) \wedge (\exists x\ [Nx] \vee \exists y\ [Py])$

Comece, como sempre, pela premissa:

1.	$\forall x\ [Nx] \wedge \exists y\ [Py]$	**P**

Agora, você pode trabalhar com a proposição inteira, como na LS:

2.	$\forall x\ [Nx]$	1 **Simp**
3.	$\exists y\ [Py]$	1 **Simp**
4.	$\forall x\ [Nx] \vee \forall y\ [Py]$	2 **Add**
5.	$\exists x\ [Nx] \vee \exists y\ [Py]$	3 **Add**
6.	$(\forall x\ [Nx] \vee \forall y\ [Py]) \wedge (\exists x\ [Nx] \vee \exists y\ [Py])$	4, 5 **Conj**

Você não pode aplicar as regras de implicação a apenas uma parte de uma proposição, mesmo que ela seja a única coisa entre colchetes em uma proposição com quantificador.

Por exemplo, observe um argumento *inválido*:

Premissas:

Alguns membros da minha família têm doutorado.

Alguns membros da minha família nunca concluíram o ensino médio.

Conclusão:

Alguns dos que têm doutorado nunca concluíram o ensino médio.

Veja a prova desse argumento:

$\exists x\ [Fx \wedge Dx], \exists x\ [Fx \wedge Nx] : \exists x\ [Dx \wedge Nx]$

1.	$\exists x\ [Fx \wedge Dx]$	**P**	
2.	$\exists x\ [Fx \wedge Nx]$	**P**	
3.	$\exists x\ [Dx]$	1 **Simp**	ERRADO!
4.	$\exists x\ [Nx]$	2 **Simp**	ERRADO!
5.	$\exists x\ [Dx \wedge Nx]$	3, 4 **Conj**	ERRADO!

Obviamente, algo está muito errado aqui. A moral da história é que você não pode aplicar a **Simp** e a **Conj**, ou qualquer outra regra de implicação àquilo que está dentro dos parênteses e ignorar o restante (mais adiante, neste capítulo, mostrarei como provar argumentos como esse, sem quebrar essa regra).

Trabalhando com formas proposicionais na LQ

No Capítulo 15, foi visto que na LQ uma constante de predicado combinada com uma variável individual (por exemplo, *Px*) não é realmente uma proposição, a menos que esteja dentro de um par de colchetes e modificada por um quantificador (por exemplo, $\forall x\ [Px]$). Quando ela aparece livre, não se trata de uma proposição, mas sim de uma *forma proposicional*.

A boa notícia é que quando você escreve provas, pode tratar as formas proposicionais como proposições. Mais adiante, neste capítulo, darei alguns exemplos específicos de como as formas proposicionais podem surgir no meio de uma prova na LQ. Por ora, basta saber que as oito regras de implicação também se aplicam às formas proposicionais.

Empregando as dez regras de equivalência na LQ

LEMBRE-SE

Você pode aplicar as dez regras de equivalência em qualquer proposição *inteira* ou forma proposicional da LQ ou a qualquer *parte* dela.

No Capítulo 10, expliquei que as regras de equivalência dão uma flexibilidade maior do que as de implicação. Uma razão para isso é que você pode aplicar as dez regras de equivalência não apenas em proposições inteiras, mas também em partes delas. Essa mesma aplicação também funciona na LQ. Por exemplo, se quiser provar o seguinte argumento:

$\exists x\ [Cx \to \sim Sx] : \exists x \sim[Cx \wedge \sim Sx]$

1.	$\exists x\ [Cx \to \sim Sx]$	**P**

Tendo em vista que $\exists x$ permanece inalterado desde a premissa até a conclusão, seu principal desafio aqui é mudar o restante da proposição. Você pode fazer isso usando duas das suas regras de equivalência favoritas:

2.	$\exists x\ [\sim Cx \vee \sim Sx]$	1 **Impl**
3.	$\exists x \sim [Cx \wedge Sx]$	2 **DeM**

No tocante às regras de equivalência, essa flexibilidade é igualmente bem aplicada nas formas proposicionais.

Por exemplo, a expressão $Cx \to \sim Sx$ é uma forma proposicional, pois a variável *x* não está sendo quantificada (veja o Capítulo 15). Você pode usar a regra de equivalência da **Impl** (veja o Capítulo 10) para escrevê-la como $\sim Cx \vee \sim Sx$.

Mais adiante, neste capítulo, discutirei como frases se transformam em provas na LQ.

Transformando Proposições com a Negação do Quantificador (NQ)

Ao transferir as regras das provas da LS para a LQ, você tem que trabalhar com os quantificadores ∀ e ∃. Em algumas provas, trabalhar os quantificadores não vai atrapalhar, mas suponha que queira provar o seguinte argumento:

$\forall x\ [Hx \to Bx] : \sim\exists x\ [Hx \wedge \sim Bx]$

Se já leu o Capítulo 16, esse argumento pode lhe parecer familiar. Lá, usei a premissa para substituir “todos os cavalos são marrons” e a conclusão para “nenhum cavalo não é marrom”. Já que essas duas proposições são equivalentes, você pode começar com a primeira e provar a segunda.

O problema aqui é que o quantificador muda de ∀ para ~∃, e nada na LS vai ajudá-lo com isso. Felizmente, a **NQ** chega para resgatá-lo. Continue lendo para ver como.

Apresentando a NQ

A negação do quantificador (**NQ**) permite trocar a proposição da LQ por uma equivalente usando os três passos a seguir:

1. **Coloque um operador ~ diante do quantificador.**
2. **Troque o quantificador (de ∀ para ∃ ou de ∃ para ∀).**
3. **Coloque um operador ~ imediatamente após do quantificador.**

Como na LS, se a proposição resultante tiver algum ocorrência de dois operadores ~ adjacentes, você poderá retirar ambos. Essa remoção usa a regra da Dupla Negação (**DN**) sem mencioná-la, como mostrei no Capítulo 10.

A Tabela 17-1 lista os quatro casos diferentes onde você pode usar a **NQ**.

Tabela 17-1 As Quatro Regras de Negação do Quantificador (NQ)		
	Proposição Direta	***Proposição Equivalente***
Todos(as)	$\forall x\ [Px]$	$\sim\exists x \sim[Px]$
Nem todos(as)	$\sim\forall x\ [Px]$	$\exists x \sim[Px]$
Algum(a)(s)	$\exists x\ [Px]$	$\sim\forall x \sim[Px]$
Nenhum(a)	$\sim\exists x\ [Px]$	$\forall x \sim[Px]$

Observe que, em cada um dos casos, o operador ~ que acompanha o quantificador não altera o que está dentro dos colchetes, simplesmente nega o conteúdo inteiro.

Assim como as dez regras de equivalência, a **NQ** funciona nas duas direções.

Por exemplo, começando por esta proposição:

$$\sim\exists x \sim[Gx \vee Hx]$$

e terminando com esta:

$$\forall x\ [Gx \vee Hx]$$

Você pode aplicar a **NQ** a uma parte da proposição, da mesma maneira que faz com as dez regras de equivalência.

Por exemplo, pode trocar a proposição a seguir:

$$\forall x\ [Mx \vee Lx] \wedge \exists x\ [Cx \vee Fx]$$

por esta:

$$\sim\exists x \sim[Mx \vee Lx] \wedge \exists x\ [Cx \vee Fx]$$

Utilizando a NQ nas provas

Usando a **NQ** e as regras da LS, você pode provar o seguinte argumento:

$\forall x\ [Bx \to Cx]$: $\sim\exists x\ [Bx \land \sim Cx]$

1.	$\forall x\ [Bx \to Cx]$	**P**

A **NQ** cuida da grande mudança envolvendo o quantificador:

2.	$\sim\exists x \sim[Bx \to Cx]$	1 **NQ**

No restante da prova, bastam alguns pequenos ajustes, usando as regras de equivalência:

3.	$\sim\exists x \sim[\sim Bx \lor Cx]$	2 **Impl**
4.	$\sim\exists x\ [Bx \land \sim Cx]$	3 **DeM**

Já que a **NQ** e as regras de equivalência são verdadeiras nas duas direções, você pode, simplesmente provar $\sim\exists x\ [Bx \land \sim Cx]$: $\forall x\ [Bx \to Cx]$ invertendo os passos. Essa inversão é a prova cabal de que essas duas proposições têm o mesmo significado, isto é, que são semanticamente equivalentes (para saber mais sobre a equivalência semântica, veja o Capítulo 6).

A Tabela 17-2 resume essa informação para as proposições do tipo *todos(as)* e *nem todos(as)*, enumerando as quatro maneiras equivalentes de escrever cada tipo de proposição.

Tabela 17-2 Quatro Maneiras Equivalentes para Escrever Proposições do Tipo Todos(as) e Nem Todos(as)

	Todos(as)	***Nem todos(as)***
Proposição Direta	$\forall x\ [Bx \to Cx]$	$\sim\forall x\ [Bx \to Cx]$

Aplicando **NQ**	$\sim\exists x \sim[Bx \rightarrow Cx]$	$\exists x \sim[Bx \rightarrow Cx]$
Aplicando a **Impl**	$\sim\exists x \sim[\sim Bx \vee Cx]$	$\exists x \sim[\sim Bx \vee Cx]$
Aplicando a **DeM**	$\sim\exists x\ [Bx \wedge \sim Cx]$	$\exists x\ [Bx \wedge \sim Cx]$

A Tabela 17-3 enumera as maneiras equivalentes para escrever as proposições do tipo *alguns(mas)* e *nenhum(a).*

Tabela 17-3 Quatro Maneiras Equivalentes para Escrever Proposições do Tipo Algum(a)(s) e Nenhum(a)

	Algum(a)(s)	***Nenhum(a)***
Proposição Direta	$\exists x\ [Bx \wedge Cx]$	$\sim\exists x\ [Bx \wedge Cx]$
Aplicando **NQ**	$\sim\forall x \sim[Bx \wedge Cx]$	$\forall x \sim[Bx \wedge Cx]$
Aplicando a **DeM**	$\sim\forall x\ [\sim Bx \vee \sim Cx]$	$\forall x\ [\sim Bx \vee \sim Cx]$
Aplicando a **Impl**	$\sim\forall x\ [Bx \rightarrow \sim Cx]$	$\forall x\ [Bx \rightarrow \sim Cx]$

Explorando as Quatro Regras de Quantificação

Quando começou a usar as regras para as provas na LS, você deve ter descoberto que regras diferentes significam utilidades diferentes. Por exemplo, a **Simp** e a **SD** eram boas para dividir as proposições, enquanto a **Conj** e a **Add** eram boas para construí-las.

Na LQ, a ideia é bem semelhante. Duas das quatro regras de quantificação são as *instanciações*, como se pode ver na Tabela 17-4. Essas regras permitem retirar o quantificador e os colchetes de uma proposição de LQ, dividindo-a para que as regras da LS possam funcionar. As outras duas são as *generalizações*. Essas permitem adicionar quantificadores e colchetes para construir as proposições necessárias para concluir a prova.

Tabela 17-4 As Quatro Regras de Quantificação na LQ e Suas Limitações

Quantificador	***Dividindo***	***Construindo***
∀	Instanciação Universal (**IU**)	Generalização Universal (**GU**)
	* Troca uma variável ligada para uma constante.	*Troca uma variável livre (não para uma livre ou para uma uma constante) por uma variável ligada.
		* Essa variável ligada não deve aparecer nas primeiras linhas em uma prova justificada por **IE** ou uma **PP** não descartada.
∃	Instanciação Existencial (**IE**)	Generalização Existencial (**GU**)
	* Muda uma variável ligada para uma livre	* Muda uma variável livre ou variável ligada.

	(não uma uma constante para uma constante).	
	* Essa variável não deve estar livre em uma linha anterior da prova.	

Duas dessas regras – **IU** e **GU** – são relativamente fáceis, então começarei por elas. A **IU** permite quebrar proposições quantificadas com ∀, enquanto a **GE** permite construir uma proposição usando o ∃.

Depois que você aprender tudo sobre **IU** e **GE**, estará pronto para a **IE** e **GU**. Essas duas regras são um pouco mais traiçoeiras, pois possuem algumas limitações que a **IU** e a **GE** não têm. Mas, a princípio, **IE** é apenas outra regra para desmembrar, e a **GU** para construir. Os tópicos a seguir trazem todas as informações necessárias para que você aprenda a usar essas quatro regras.

Regra fácil nº 1: Instanciação universal (IU)

LEMBRE-SE

A instanciação universal (**IU**) permite:

- ✓ Liberar a variável ligada a uma proposição ∀, removendo o quantificador e os colchetes.
- ✓ Para, depois, trocar essa variável uniformemente em todos os lugares em que ela aparece por qualquer constante individual ou variável que você quiser.

Por exemplo, sabendo que esta proposição é verdadeira:

Todas as cobras são répteis.

Na LQ, você pode expressar esta proposição como

$\forall x\ [Sx \rightarrow Rx]$

Tendo em vista que você sabe um fato sobre *todas* as cobras, é seguro afirmar que uma proposição semelhante a respeito de uma cobra *específica* também será verdadeira:

Se Binky é uma cobra, então Binky é um réptil.

A **IU** permite dar esse salto, formalmente:

$Sb \to Rb$

O resultado é uma proposição que se parece muito com uma da LS, o que significa que você pode usar as 18 regras da LS.

Aquecendo para as provas

Para praticar, este tópico traz um exemplo de uma prova para você começar a se aquecer. Se quiser provar a validade deste argumento:

Premissas:

Todos os elefantes são cinzas.

Tiny é um elefante.

Conclusão:

Tiny é cinza.

Traduza esse argumento para a LQ, assim:

$\forall x\ [Ex \to Gx], Et : Gt$

Como na prova da LS, comece por enumerar as premissas:

1.	$\forall x\ [Ex \to Gx]$	**P**
2.	Et	**P**

Agora, "desembrulhe" a primeira premissa usando a **IU**:

3.	$Et \to Gt$	1 **IU**

Aqui, primeiro removi o $\forall x$ e os colchetes. Depois, optei por mudar a variável *x* uniformemente (isto é, onde quer que ela apareça) para uma constante individual *t*. Usarei *t* nesse caso, pois essa constante aparece na linha 2, o que será útil no próximo passo.

Agora, você pode concluir a prova usando uma velha e conhecida fórmula da LS:

4.	Gt	2, 3 **MP**

Usos válidos e inválidos da IU

A **IU** lhe dá uma série de opções quando você está procurando os próximos passos da prova.

Por exemplo, considerando a seguinte premissa:

$\forall x\ [(Pa \wedge Qx) \rightarrow (Rx \wedge Sb)]$

A coisa mais simples que a **IU** permite fazer é remover o quantificador e os colchetes e manter a mesma variável:

$(Pa \wedge Qx) \rightarrow (Rx \wedge Sb)$

A **IU** também permite mudar a variável, desde que essa alteração seja feita uniformemente na expressão inteira:

$(Pa \wedge Qy) \rightarrow (Ry \wedge Sb)$

Além do mais, com a **IU** você troca a variável para qualquer constante que quiser, mesmo aquelas que já aparecem na proposição, desde que faça a troca uniformemente na proposição. Por exemplo:

$(Pa \wedge Qa) \rightarrow (Ra \wedge Sb)$

$(Pa \wedge Qb) \rightarrow (Rb \wedge Sb)$

$(Pa \wedge Qc) \rightarrow (Rc \wedge Sb)$

Em cada uma das hipóteses, substituí a variável *x* uniformemente por uma constante — primeiro *a*, depois *b* e *c*.

> Quando usa a **IU**, você pode substituir a variável uniformemente com uma única escolha de constante ou variável; além disso, deve preservar as constantes da proposição original.

Por exemplo, observe um uso *inválido* da **IU**:

$(Pa \wedge Qa) \rightarrow (Rb \wedge Sb)$ ERRADO!

Nessa proposição inválida, substituí incorretamente um *x* por um *a*, e o outro por *b*. Essas substituições estão erradas pois as trocas devem ser uniformes — você deve substituir todos os *x* pela mesma constante ou pela mesma variável. Observe outro uso inválido da **IU:**

$(Px \wedge Qx) \rightarrow (Rx \wedge Sx)$ ERRADO!

Nessa proposição, alterei incorretamente as constantes *a* e *b*. Essas substituições estão erradas, pois a **IU** somente permite trocar a variável que está sendo quantificada na proposição original, nesse caso, *x*. As outras variáveis e constantes devem permanecer iguais.

Regra fácil nº 2: Generalização Existencial (GE)

A generalização existencial (**GE**) lhe permite mudar uma constante individual ou uma variável livre para uma variável ligada adicionando colchetes e quantificando-a com o $\exists$.

Por exemplo, sabendo que esta proposição é verdadeira:

Meu carro é branco.

Na LQ, você pode expressar essa proposição como

Wc

Tendo um exemplo de uma coisa específica que é branca, você sabe, de forma mais geral, que *algo* é branco, sendo seguro dizer que

Existe um x tal que x é branco.

Pode formalizá-la assim:

$\exists x\ [Wx]$

Aquecendo com uma prova

Assim como a **IU** lhe permite dividir as proposições da LQ no início de uma prova, a **GE** permite reconstruí-las no final.

Por exemplo, se quiser provar que o seguinte argumento é válido:

$\forall x\ [Px \rightarrow Qx],\ \forall x\ [(Pb \wedge Qx) \rightarrow \mathrm{R}x],\ Pa \wedge Pb : \exists x\ [Rx]$

1.	$\forall x\ [Px \rightarrow Qx]$	**P**
2.	$\forall x\ [(Pb \wedge Qx) \rightarrow Rx]$	**P**
3.	$Pa \wedge Pb$	**P**

Primeiro, use a **IU** para desembrulhar as primeiras duas premissas:

4.	$Pa \rightarrow Qa$	1 **IU**
5.	$(Pb \wedge Qa) \rightarrow Ra$	2 **IU**

A linha 5 ilustra o ponto que foi discutido no tópico anterior: você pode substituir todos as variáveis x na proposição 2 pela constante a, mas a constante b deve permanecer a mesma.

Depois, use a **Simp,** para quebrar a proposição 3:

6.	Pa	3 **Simp**
7.	Pb	3 **Simp**

Agora você pode usar seus velhos truques das provas na LS:

8.	Qa	4, 6 **MP**

9. $Pb \wedge Qa$ 7, 8 **Conj**

10. Ra 5, 9 **MP**

Você descobriu agora um exemplo da constante (*a*) que tem a propriedade que está procurando (*R*) e, com isso, conclui a prova, usando a **GE**:

11. $\exists x\,[Rx]$ 10 **GE**

Usos válidos e inválidos da GE

A **GE** lhe dá as mesmas opções que a **IU**, mas com ela você está adicionando colchetes e ligando variáveis, ao invés de removendo os colchetes e liberando as variáveis.

A **GE** permite mudar qualquer constante para uma variável e, então, ligar essa variável usando o quantificador $\exists$. Assim como na **IU**, as mudanças devem ser uniformes em toda a proposição. Por exemplo, considere esta premissa:

$$(Pa \wedge Qb) \rightarrow (Rb \vee Sa)$$

Você pode usar a **GE** para mudar qualquer constante (*a* ou *b*) para uma variável e ligá-la a um qualificador $\exists$:

$$\exists x\,[(Px \wedge Qb) \rightarrow (Rb \vee Sx)]$$

$$\exists x\,[(Pa \wedge Qx) \rightarrow (Rx \vee Sa)]$$

Assim como na **IU**, quando você usa a **GE**, as trocas de constantes devem ser feitas uniformemente. Por exemplo, veja três usos inválidos da **GE**:

$\exists x\,[(Px \wedge Qx) \rightarrow (Rx \vee Sx)]$ ERRADO!

Nessa proposição inválida, troquei incorretamente as constantes *a* e *b* para *x*.

$\exists x\ [(Px \wedge Qb) \rightarrow (Rb \vee Sa)]$ ERRADO!

Aqui, fiz diferente e troquei um *a* e não o outro.

$\exists x\ [(Pa \wedge Qb) \rightarrow (Rb \vee Sa)]$ ERRADO!

Nessa proposição, não troquei *nenhuma* constante por variável.

A **GE** ainda permite que você faça alterações semelhantes em formas proposicionais. Estas não serão premissas em um argumento, mas poderão surgir do uso da **IU** e de outras regras de quantificação. Considere a seguinte forma proposicional:

$(Vx \wedge Ty) \vee (Cx \rightarrow Gy)$

Você pode usar a **GE** para ligar ambas as variáveis:

$\exists x\ [(Vx \wedge Ty) \vee (Cx \rightarrow Gy)]$

$\exists y\ [(Vx \wedge Ty) \vee (Cx \rightarrow Gy)]$

Pode, também, usar a **GE** para trocar uma variável, desde que o faça de modo uniforme na proposição inteira:

$\exists z\ [(Vx \wedge Tz) \vee (Cx \rightarrow Gz)]$

$\exists z\ [(Vz \wedge Ty) \vee (Cz \rightarrow Gy)]$

Observe que os quatro exemplos são também formas proposicionais, pois em cada um há uma variável livre.

Assim como ocorre com as constantes, ao usar a **GE** para ligar ou trocar variáveis, as alterações devem ser uniformes na forma proposicional inteira. Veja alguns usos inválidos da **GE** aplicada às formas proposicionais:

$\exists x\ [(Vx \wedge Tx) \vee (Cx \rightarrow Gx)]$ ERRADO!

Nessa proposição inválida, usei incorretamente a **GE** para ligar as variáveis *x* e *y*, usando a variável *x.*

$\exists z\ [(Vx \wedge Tz) \vee (Cx \rightarrow Gy)]$ ERRADO!

Aqui, troquei um *y* por um *z*, mas não o outro *y*.

$\exists z\ [(Vx \wedge Ty) \vee (Cx \rightarrow Gy)]$ ERRADO!

Nessa proposição, liguei a variável *z*, mas não troquei *nenhuma* variável por *z.*

Regra não tão fácil nº 1: Instanciação Existencial (IE)

A instanciação existencial (**IE**) lhe permite liberar uma variável ligada em uma proposição $\exists$ (ou trocá-la por uma variável diferente), removendo o quantificador e os colchetes, *desde que* essa variável já não esteja livre em uma linha anterior da prova.

Você está certo, essa descrição é muito complicada. Então, para facilitar, explico passo a passo. Começarei esclarecendo tudo sobre a expressão *desde que*.

Em que a IE se parece com a IU

A primeira parte da descrição da **IE** é semelhante à da **IU**, exceto que ela funciona com proposições $\exists$ (ao invés de $\forall$). Considere a seguinte proposição:

Algo é verde.

Você pode representar essa proposição em LQ assim:

$\exists x\ [Gx]$

Como na **IU**, a **IE** lhe permite liberar a variável ligada, ficando assim:

Gx

Essa forma proposicional significa *"x* é verde", que é parecido com o significado da proposição original. A **IE** também lhe permite trocar a variável:

Gy

Nesse caso, a forma proposicional significa "*y* é verde", o que é bem parecido com a forma original, pois as variáveis são apenas símbolos.

A IE funciona com variáveis, mas não com constantes

Ao contrário da **IU**, porém, a **IE** não lhe permite trocar a variável pela constante. Por exemplo, considere o seguinte:

1.	$\exists x\ [Gx]$	**P**	
2.	Ga	1 **IE**	ERRADO!

O motivo pelo qual você não pode usar a **IE** nesse caminho é simples: o fato de a frase original ter dito que *algo* é verde não quer dizer que você pode presumir que molho de maçã, tatus, Alasca, Ava Gardner ou qualquer outra coisa *específica* que *a* represente seja também verde.

A **IE** é mais restritiva do que a **IU**. Com a **IU**, você começa sabendo que *tudo* tem determinada propriedade, para concluir que qualquer constante ou variável também a tem. Com a **IE**, começa sabendo apenas que *algo* tem aquela propriedade e, assim, só pode concluir que uma variável também a tem.

Um exemplo usando a IE

Para ajudar a entender melhor como usar a **IE**, considere o seguinte argumento:

$\forall x\ [Cx \rightarrow Dx], \exists x\ [\sim Dx] : \sim\forall x\ [Cx]$

1.	$\forall x\ [Cx \rightarrow Dx]$	**P**
2.	$\exists x\ [\sim Dx]$	**P**

Primeiro, use a **IU** e a **IE** para remover os quantificadores. Mas, use a **IE** *antes* da **IU** (essa ordem tem a ver com a definição da **IE** que vem após a palavra *desde que*, o que explicarei depois).

3.	$\sim Dx$	2 **IE**
4.	$Cx \rightarrow Dx$	3 **IU**

O próximo passo é óbvio, mesmo que você ainda não tenha certeza de como lhe será útil:

5. $\sim Cx$ 3, 4 **MT**

Agora, você sabe algo sobre a variável *x* — mais especificamente, que *x* não tem a propriedade *C*. Então, pode usar a **GE** para fazer uma proposição mais geral “existe um *x* tal que *x* não tem a propriedade *C*”:

6. $\exists x\ [\sim Cx]$

Para concluir a prova, use a **NQ**. Para que fique claro, dê uma passo extra e use, explicitamente, a Dupla Negação (**DN**):

7. $\sim\forall x \sim[\sim Cx]$ 6 **NQ**

8. $\sim\forall x\ [Cx]$ 7 **DN**

A IE somente permite a liberação de uma variável que não esteja livre em outro local

No exemplo do tópico anterior, usei, propositadamente, a **IE** antes da **IU**. Agora explicarei o porquê.

Observe que a última parte da definição da **IE** diz “*desde que* esta variável já não esteja livre em uma linha anterior da prova”. Então, no exemplo do tópico anterior, se eu tivesse escrito $Cx \rightarrow Dx$ na linha 3, não poderia usar a **IE** para escrever $\sim Dx$ na linha 4.

Essa limitação pode parecer uma tecnicalidade, mas, acredite, é importante. Para demonstrar o porquê, simularei a prova de um argumento nitidamente falso. Então, mostrarei onde surge o problema. Veja este argumento:

Premissas:

Pessoas existem.

Gatos existem.

Conclusão:

Algumas pessoas são gatos.

O argumento traduzido para LQ é:

$\exists x\ [Px], \exists x\ [Cx] : \exists x\ [Px \wedge Cx]$

E aqui está a “prova”:

1.	$\exists x\ [Px]$	**P**	
2.	$\exists x\ [Cx]$	**P**	
3.	Px	1 **IE**	
4.	Cx	2 **IE**	ERRADO!
5.	$Px \wedge Cx$	3, 4 **Conj**	
6.	$\exists x\ [Px \wedge Cx]$	5 **GE**	

Algo deve estar errado, caso contrário, você encontraria pessoas-gato perambulando por aí (fico pensando se elas se engasgariam com bolas de pelos). Então, onde está o problema?

A linha 3 está correta: uso a **IE** para liberar a variável *x*. Mas, na linha 4, tento usar a **IE** para liberar a variável *x* outra vez. Esse movimento é inaceitável e me leva a arruinar o resultado. *Já* tinha liberado a variável *x* na linha 3, por isso não tenho permissão para usar essa variável com a **IE** na linha 4.

Eu poderia ter escrito *Py* ou *Pz* na linha 4, o que faria com que o restante do argumento desmoronasse. Isso seria bom, pois argumentos ruins *devem* mesmo desmoronar.

Usos válidos e inválidos da IE

Neste tópico, esclareço quando e onde exatamente você pode usar a **IE.**

Por exemplo, considere estas duas premissas:

1. $\exists x\ [(Rx \wedge Fa) \rightarrow (Hx \wedge Fb)$ **P**

2. $\exists y\ [Ny]$ **P**

Como na **IU**, a **IE** lhe permite liberar uma variável, com a opção de trocá-la no caminho. Veja três passos válidos:

3. $(Rx \wedge Fa) \rightarrow (Hx \wedge Fb)$ 1 **IE**

4. $(Ry \wedge Fa) \rightarrow (Hy \wedge Fb)$ 1 **IE**

5. Nz 2 **IE**

Na linha 3, usei a **IE** para liberar a variável *x*. Esse é o uso mais comum da **IE**. Na linha 4, troquei a variável de *x* para *y*, e a liberei usando a **IE**. Essa é uma opção válida. Então, na linha 5, optei pela **IE** para liberar a variável da linha 2, mas, porque *já* tinha liberado *x* e *y* nas linhas anteriores, precisei escolher uma nova variável, *z.*

Como acontece na **IU**, se você trocar a variável usando a **IE**, terá que fazer de modo uniforme. Por exemplo, veja esta proposição inválida:

6. $(Rv \wedge Fa) \rightarrow (Hw \wedge Fb)$ 1 **IE** ERRADO!

Na linha 6, troquei incorretamente *x* por *v* e outro por *w*. Essa troca é tão errada com a **IE** quanto era com a **IU**.

A **IE**, porém, limita suas opções em aspectos que a **IU** não faz. Por exemplo, observe esta proposição inválida:

7. $(Ra \wedge Fa) \rightarrow (Ha \wedge Fb)$ 1 **IE** ERRADO!

Na linha 7, troquei uniformemente a variável *x* por uma constante individual *a.* Isso é proibido! Com a **IE**, as variáveis devem permanecer variáveis. Observe esta última proposição inválida:

8. Nx 2 **IE** ERRADO!

Na linha 8, mudei a variável *y* para *x*. Mas a variável *x* já aparece como uma variável livre na linha 3, então você não pode usá-la uma segunda vez com a **IE**. Esse tipo de erro é o mais comum na **IE**, tenha cuidado.

Regra não tão fácil nº 2: Generalização Universal (GU)

A Generalização Universal (**GU**) permite que você troque uma variável livre por uma ligada, adicionando colchetes e quantificando-a com ∀, *desde que* essa variável não esteja livre em uma linha anterior da prova que tenha sido justificada pela **IE** ou por uma PP não descartada (veja o Capítulo 11 para saber mais sobre descarte da **PP**).

Como na **IE**, a **GU** é bem direta até que surge a expressão *desde que.* Para facilitar ao máximo, explicarei isso passo a passo.

Em que a GU é parecida com a GE

A **GU** é muito parecida com a **GE**. A maior diferença é que a **GE** trabalha com proposições ∃ (ao invés de ∀, usadas pela **GU**). Veja a seguinte definição:

Considere *Nx* = *x* é agradável.

Presuma que em algum ponto da prova você chegue na seguinte forma proposicional:

Nx

Em circunstâncias apropriadas, a **GU** lhe permite chegar à proposição mais geral “qualquer que seja *x*, *x* é agradável” ou simplesmente “tudo é agradável”:

∀*x* [*Nx*]

A variável escolhida nesse caso não importa, você pode, ainda, usar a **GU** para escrever:

∀*y* [*Ny*]

A GU trabalha com variáveis, mas não com constantes

Ao contrário da **GE**, a **GU** não lhe permite trocar uma constante por uma variável. Por exemplo, considere o seguinte:

1.	Nc	**P**	
2.	$\forall x\ [Nx]$	1 **GU**	ERRADO!

A razão de essa proposição estar errada é muito simples: o fato de a frase original dizer que *c* (seja lá o que ele estiver representando: carros, vacas, Cadillacs ou Cameron Diaz) é agradável não permite que se conclua que *tudo* também é.

A **GU** é mais restritiva do que a **GE**. Com a **GE**, você quer demonstrar que *algo* tem uma determinada propriedade, encontrando uma coisa específica — uma constante ou uma variável —, que tenha aquela propriedade. Com a **GU**, porém, quer demonstrar que *tudo* tem aquela propriedade, encontrando uma variável que a possua.

Um exemplo usando a GU

Assim como na **GE**, a **GU** é, em geral, mais utilizada no final da prova para transformar uma forma proposicional em proposição, usando o quantificador ∀. Considere este argumento:

$\forall x\ [Bx \to Cx],\ \forall y\ [\sim Cy \vee Dy] : \forall x\ [Bx \to Dx]$

1.	$\forall x\ [Bx \to Cx]$	**P**
2.	$\forall y\ [\sim Cy \vee Dy]$	**P**

O primeiro passo é abrir as duas premissas usando a **IU**:

3.	$Bx \to Cx$	1 **IU**
4.	$\sim Cx \vee Dx$	2 **IU**

Observe que na linha 4, a **IU** lhe dá a liberdade de trocar a variável *y* por *x*, sem a preocupação de que *x* já esteja livre na linha 3. Uma vez que as variáveis são todas iguais, você pode prosseguir como se tivesse escrevendo uma prova de LS:

5.	$Cx \to Dx$	4 **Impl**
6.	$Bx \to Dx$	3, 5 **SH**

Agora, tudo está montado para completar a prova com a **GU**:

7.	$\forall x\, [Bx \to Dx]$	6 **GU**

A GU somente permite a ligação de uma variável que não esteja livre em uma linha justificada pela IE ou por uma PP não descartada

Vou confessar uma coisa: acho que a regra da **GU** é o conceito mais complicado deste livro. Pronto, falei! O que torna a **GU** tão problemática é essa regra sobre como ela *não* deve ser usada.

Mas, veja pelo lado bom: quando aprender a **GU**, você pode estar certo de que nada pior está por vir.

Imagine, a princípio, que fundou um clube de um único membro: você. Mesmo começando pequeno, você tem grandes planos para o clube: quer que algum dia todo mundo entre para esse clube. Seu senso de lógica anda um pouco distorcido e você constrói o seguinte argumento:

Premissa:

Um membro do meu clube existe.

Conclusão:

Tudo é um membro do meu clube.

Então, você traduz esse argumento para LQ assim:

$\exists x\, [Mx] : \forall x\, [Mx]$

Isso parece um pouco improvável, mas você “prova” seu argumento assim:

1.	$\exists x\ [Mx]$	**P**	
2.	Mx	1 **IE**	
3.	$\forall x\ [Mx]$	2 **GU**	ERRADO!

De alguma forma, em dois passos fáceis, você ampliou o conjunto de membros de seu clube de apenas um membro para tudo o que existe no universo. Imagine o que a UNICEF poderia fazer com esse argumento!

Veja o problema: a variável *x* está livre na linha 2, que é justificada por **IE**, de forma que você não pode usar a **GU** para ligar *x* na linha 3. Esse cenário é exatamente o que a definição da **GU** depois da expressão *desde que* está querendo dizer.

Como outro exemplo, imagine que você *não* é um bilionário (se isso for difícil de imaginar, repita comigo: “eu não sou um bilionário”). A partir dessa premissa, você tentará provar que Donald Trump não é um bilionário. Veja o seguinte argumento:

Premissa:

Eu não sou um bilionário.

Conclusão:

Donald Trump não é bilionário.

A tradução desse argumento para LQ é a seguinte:

$\sim Bi : \sim Bt$

E, agora, veja a “prova”:

1.	$\sim Bi$	**P**
2.	Bx	**PP**

Aqui, uso uma estratégia avançada de prova indireta que foi mais discutida no Capítulo 12: presumi um premissa (**PP**) e trabalhei para provar sua contradição. Se conseguir, terei provado a *negação* da **PP**. Em geral, essa é uma estratégia perfeitamente válida, mas o próximo passo contém um erro fatal:

3.	$\forall x\ [Bx]$	1 **GU**	ERRADO!
4.	Bi	2 **IU**	
5.	$Bi \wedge \sim Bi$	1, 4 **Conj**	
6.	$\sim Bx$	2-5 **PI**	

Tendo descartado minha **PP**, o resto da prova parece simples:

7.	$\forall x\ [\sim Bx]$	6 **GU**
8.	$\sim Bt$	7 **IU**

O problema dessa vez é que a variável *x* está livre na linha 2, que é justificada por uma **PP** *não descartada*, o que significa que você não pode usar a **GU** para ligar *x.*

Observe, porém, que na linha 7, a prova inclui um uso perfeitamente válido da **GU** para ligar *x*. A única diferença aqui, é que a essa altura da prova a **PP** já foi descartada.

A moral da história é que sempre que você quiser usar a **GU**, precisará verificar primeiro se a variável que está tentando ligar não está livre em nenhuma destas circunstâncias:

- ✓ Uma linha que é justificada pela **IE**
- ✓ Uma linha que é justificada por uma **PP** que *não* tenha sido descartada.

Usos Inválidos e Válidos da GU

A seguir está um resumo de quando você pode ou não usar a **GU**. Suponha que esteja fazendo a seguinte prova (essa precisa de um pouco de preparação):

1.	$\forall x\ [Tx]$	**P**
2.	$\exists y\ [Gy]$	**P**
3.	Sa	**P**
4.	Tx	1 **IU**
5.	Gy	2 **IE**
6.	$Tx \wedge Sa$	3, 4 **Conj**
7.	$(Tx \wedge Sa) \wedge Gy$	5, 6 **Conj**
8.	Hz	**PP**

Agora, você pode usar a **GU** para ligar a variável *x* na linha 7:

9. $\forall x\ [(Tx \wedge Sa) \wedge Gy]$ 7 **GU**

Mas, *não pode* usar a **GU** para trocar a constante *a* na linha 7 por uma variável e ligá-la:

10. $\forall w\ [(Tx \wedge Sa) \wedge Gy]$ 7 **GU** ERRADO!

Também *não pode* usar a **GU** para ligar a variável *y* na linha 7, pois essa variável aparece livre na linha 5, que é justificada pela **IE**:

11. $\forall y\ [(Tx \wedge Sa) \wedge Gy]$ 7 **GU** ERRADO!

Além do mais, *não pode* usar a **GU** para ligar a variável *z* na linha 8, pois essa variável é justificada por uma **PP** que ainda não foi descartada:

12. $\forall z\ [Hz]$ 8 **GU** ERRADO!

Capítulo 18

Boas Relações e Identidades Positivas

Neste Capítulo

- Entendendo as expressões relacionais
- Descobrindo identidades
- Escrevendo provas usando relações e identidades

Na Lógica Sentencial (LS), você representa uma proposição básica com apenas uma letra — uma constante. Por exemplo, pode representar a proposição "Bob é jardineiro" como

B (TRADUÇÃO PARA LS)

A mesma frase é representada na Lógica Quantitativa (LQ) com uma combinação de duas letras, uma constante de predicado e uma constante individual, por exemplo:

Tb

Essa representação funciona muito bem. Mas se quiser traduzir a proposição "Bob contratou Marty" para LQ, precisará de um modo de expressar a relação entre essas pessoas, já que agora há mais de uma em cena.

Se quiser traduzir a proposição "Bob é o melhor jardineiro do país", não estará descrevendo uma propriedade que se aplica a Bob, mas sim sua identidade como o *único* melhor jardineiro do país.

Felizmente, a LQ permite que você expresse facilmente essas ideias mais sofisticadas. Neste capítulo, discuto as relações e as identidades na LQ. *Relações* permitem que você expresse proposições que têm mais do que um ator principal, e as *identidades* o ajudam em situações onde o ator principal é identificado por uma descrição alternativa que lhe é exclusiva. Por fim, mostrarei como esses novos conceitos se encaixam na LQ.

Relacionando com as Relações

Até agora você usou *expressões monadárias*: proposições e formas proposicionais que só possuem uma constante ou uma variável individual. Por exemplo, a proposição

$$Na$$

só tem uma constante individual: a. Do mesmo modo a forma proposicional

$$Nx$$

tem apenas uma variável: x.

Neste tópico, o ajudarei a expandir sua noção de expressão para incluir aquelas com mais de uma constante ou variável individual.

Definindo e usando relações

Uma *expressão relacional* tem *mais de uma* constante ou variável individual.

Definir uma expressão relacional é bem mais fácil depois que você sabe o básico sobre expressão monadária (veja o Capítulo 15). Uma expressão monadária pode ser definida da seguinte forma:

Considere Nx = x é barulhento

Então você pode traduzir a frase “AuAu é barulhento” como

$$Na$$

Mas suponha que tenha a proposição:

AuAu é mais barulhento do que Bóris.

Nesse caso você pode começar pela definição da seguinte expressão relacional:

Considere Nxy = x é mais barulhento que y

Assim, use essa expressão para traduzir a proposição da seguinte forma:

Nab

Da mesma maneira você pode traduzir a proposição

Bóris é mais barulhento do que AuAu

como

Nba

A ordem das constantes ou variáveis individuais em uma expressão relacional é crucial — não as inverta!

Conectando expressões relacionais

Você pode usar os cinco operadores da LS (~, ∧, ∨, → e ↔) para conectar proposições e formas proposicionais relacionais da mesma maneira que nas expressões monadárias (para saber mais sobre como distinguir formas proposicionais de proposições, veja o Capítulo 15).

Depois que você traduz uma proposição do português para uma expressão relacional, ela se torna uma unidade indivisível, assim como a expressão monadária. A partir daí, as mesmas regras se aplicam no uso dos cinco operadores.

Por exemplo, se quiser dizer:

Kátia é mais alta do que Cristiano, mas Cristiano é mais alto do que Paula.

Primeiro, defina as constantes:

Considere *k* = Kátia

Considere *c* = Cristiano

Considere *p* = Paula

Depois, defina a expressão relacional adequada:

> Considere Txy = x é mais alta que y

Agora você pode traduzir a proposição

> $Tkc \wedge Tcp$

Você pode igualmente traduzir a proposição

> Se Kátia é mais alta do que Cristiano, então ela é também mais alta do que Paula.

como

> $Tkc \rightarrow Tkp$

Usando os quantificadores nas relações

Os dois quantificadores ($\forall$ e $\exists$) podem ser usados para conectar proposições e formas proposicionais relacionais, assim como nas expressões monadárias.

LEMBRE-SE

Usar os quantificadores nas relações não é muito diferente de usá-los nas expressões monadárias.

Por exemplo, se quiser traduzir a seguinte proposição do português para a LQ:

> Todos gostam de chocolate.

Primeiro, defina a expressão relacional:

Considere Lxy = x gosta de y

Então, defina a constante individual:

> Considere c = chocolate

Agora está pronto para traduzir a proposição, assim:

> $\forall x\ [Lxc]$

Da mesma forma, considere a seguinte expressão:

Alguém apresentou Diana a Joel.

Desta vez você precisa definir uma expressão relacional que contenha três variáveis individuais:

Considere $Ixyz$ = x apresentou y a z

Usando as iniciais de Diana e Joel como constantes individuais e ligando a variável x com o quantificador $\exists$, você pode traduzir a proposição:

$\exists x\ [Ixdj]$

Pode também traduzir a proposição como

Joel apresentou Diana para todos.

como

$\forall x\ [Ijdx]$

Observe que você ainda pode usar a variável x aqui, embora a definição da expressão relacional use y nessa posição. A escolha da variável não é importante, desde que ela tenha sido quantificada adequadamente.

Trabalhando com múltiplos quantificadores

Já que as relações têm mais de uma variável, elas abrem a possibilidade de proposições com mais de um quantificador. Por exemplo, você pode traduzir a proposição

Todo mundo apresentou alguém para Joel.

como

$\forall x\ \exists y\ [Ixyj]$

Ao desmembrar essa proposição, você percebe que ela declara literalmente que: “qualquer que seja x, existe um y tal que x apresentou y para Joel”.

Da mesma forma, se quiser traduzir a seguinte proposição para a LQ:

Joel apresentou alguém para todos.

Pode usar a seguinte proposição:

$\forall x \, \exists x \, [Ijyx]$

Essa proposição significa que "Qualquer que seja *x*, existe um *y* tal que Joel apresentou *y* para *x".*

Tenha cuidado com a ordem dos quantificadores, pois alterá-la de $\forall x \, \exists y$ para $\exists x \, \forall x$ muda o significado, mesmo se o conteúdo dos colchetes permanecer o mesmo.

Veja as distinções entre proposições iniciadas com $\forall x \, \exists y$ e aquelas começadas com $\exists y \, \forall x$

- ✓ De modo geral, $\forall x \, \exists y$ significa "qualquer que seja *x*, existe um *y* tal que...", o que significa que *y pode ser diferente* para dois *x* diferentes.
- ✓ Por outro lado, $\exists y \, \forall x$ significa "existe um *y* tal que qualquer que seja *x...",* o que significa que o *y é o mesmo* para todo *x.*

Para deixar clara essa distinção, defino uma nova constante de predicado:

Considere *Mxy* = *x* é casado com *y*

Agora, se você quiser expressar a ideia de que todas as pessoas são casadas, pode traduzir a proposição como

$\forall x \, \exists y \, [Mxy]$

Essa proposição literalmente significa "para todo *x*, existe um *y* tal que *x* é casado com *y*".

Mas suponha que eu inverta os quantificadores:

$\exists y \, \forall x \, [Mxy]$

Agora ela quer dizer que "existe um *y* que, para todo *x*, *x* é casado com *y*". Nesse caso, estou dizendo que todas as pessoas são casadas com *a mesma pessoa*. Isso definitivamente não é o que eu tinha em mente.

Quantificadores múltiplos são especialmente úteis para expressar ideias matemáticas. Por exemplo, se você quiser expressar a ideia de que os números cardinais (1, 2, 3...) são infinitos, pode manifestá-la definindo a expressão relacional:

Considere Gxy = x é maior do que y

Agora pode expressar a infinitude de todos os números declarando que para cada número existe outro que é maior:

$\forall x \exists y\ [Gyx]$

Em outras palavras, "qualquer que seja *x*, existe um *y* tal que *y* é maior do que *x*".

Escrevendo provas com relações

Quando você escreve as provas, as expressões relacionais funcionam quase da mesma forma que as monadárias. Elas permanecem como porções distintas e inseparáveis que você pode manipular usando as 18 regras de inferência da LS (veja os Capítulos 9 e 10).

A Negação do Quantificador (**NQ**) e as quatro regras de quantificação também funcionam da mesma forma com as expressões relacionais e as monadárias (esses fabulosos resultados lógicos foram vistos no Capítulo 17). A principal diferença é que agora você talvez tenha que incluir proposições com quantificadores múltiplos.

Usando NQ com múltiplos quantificadores

A NQ funciona do mesmo modo, tanto com quantificadores múltiplos quanto com únicos (veja o Capítulo 17 para uma revisão sobre o uso da **NQ**). Apenas certifique-se de qual é o quantificador que está sendo modificado.

Por exemplo, veja esta nova declaração:

Considere *Pxy* = *x* deu um presente a *y*

Se quiser provar o seguinte argumento:

Premissa:

Todos deram presente pelo menos para uma pessoa.

Conclusão:

Ninguém não deu um presente para ninguém.

Você pode escrever esse argumento em LQ da seguinte forma:

$\forall x \, \exists y \, [Pxy] : \sim\exists x \, \forall y \sim[Pxy]$

1.	$\forall x \, \exists y \, [Pxy]$	**P**
2.	$\sim\underline{\exists x} \sim\exists y \, [Pxy]$	1 **NQ**

Na linha 2 sublinhei o primeiro quantificador, pois ele é o centro da alteração na linha 1. Observe que troquei o operador ~ antes do quantificador, modifiquei-o de $\forall x$ para $\exists x$ e inseri outro operador ~ depois do quantificador.

3.	$\sim\exists x \sim\sim \underline{\forall y} \sim[Pxy]$	2 **NQ**
4.	$\sim\exists x \, \underline{\forall y} \sim[Pxy]$	3 **DN**

Na linha 3, o segundo quantificador é agora o foco da mudança da linha 2. Mais uma vez adicionei o operador ~ antes e depois desse quantificador e troquei $\exists y$ por $\forall y$. Para manter a prova mais clara, realizei uma linha adicional para aplicar a regra da Dupla Negação (**DN**).

Usando as quatro regras de quantificação com múltiplos quantificadores

Quando usa as quatro regras de quantificação — **IU**, **GE**, **IE** e **GU** (veja o Capítulo 17) — com múltiplos quantificadores, uma restrição importante entra em jogo:

você pode apenas remover o primeiro quantificador (o mais à esquerda) ou adicionar um novo primeiro quantificador.

Na prática, essa restrição significa que você tem que fazer o seguinte:

- ✓ Dividir proposições de fora para dentro.
- ✓ Construir proposições de dentro para fora.

Por exemplo, observe esta prova:

$$\forall x\,\forall y\,[Jxy \rightarrow Kyx],\ \exists x\,{\sim}\exists y\,[Kyx] : \exists x\,\forall y\,[{\sim}Jxy]$$

1.	$\forall x\,\forall y\,[Jxy \rightarrow Kyx]$	**P**
2.	$\exists x\,{\sim}\,\exists y\,[Kyx]$	**P**

Primeiro, uma rápida troca de quantificadores na segunda premissa para deslocar o operador ~ para a direita e tirá-lo do caminho:

3.	$\exists x\,\forall y\,{\sim}\,[Kyx]$	2 **NQ**

Mas essa premissa ainda tem um quantificador ∃, de forma que você tem que usar a instanciação existencial (**IE**). Preciso dar esse passo o mais breve possível, antes que o *x* apareça livre na prova (veja o Capítulo 17 para saber mais sobre **IE**). Então, agora a desmembro de fora para dentro:

4.	$\forall y\,{\sim}[Kyx]$	3 **IE**
5.	${\sim}Kyx$	4 **IU**

Agora estou na primeira premissa e de novo continuo desmembrando, de fora para dentro:

6.	$\forall y\,[Jxy \rightarrow Kyx]$	1 **IU**

7. $Jxy \to Kyx$ 6 **IU**

O próximo passo é muito claro:

8. $\sim Jxy$ 5, 7 **MT**

É hora de construir a conclusão, de dentro para fora:

9. $\forall y\ [\sim Jxy]$ 8 **GU**

Considerações sobre as proposições autorreferenciais

Alguns pontos mais delicados nas provas com relações estão além do objeto deste livro. Muitas dessas questões surgem das *proposições autorreferenciais*, que são expressões relacionais com uma constante individual repetida.

Por exemplo, quando você tem a proposição “Joel ama todo mundo” — $\forall x\ [Ljx]$ —, pode usar a *IU* para inferir a proposição “Joel ama a si mesmo “— Ljj.

Proposições autorreferenciais são necessárias para tornar a LQ totalmente expressiva, mas você tem que tomar cuidados especiais com elas nas provas.

Esse uso da generalização universal (**GU**) é válido pois na linha 4 a variável *y* ainda está ligada. Finalmente, você usa a generalização existencial (**GE**) para concluir a prova:

$\exists x\ \forall x\ [\sim Jxy]$ 9 **GE**

Identificando com as Identidades

Observe as duas proposições a seguir:

> Prudente de Morais foi presidente do Brasil.
>
> Prudente de Morais foi o primeiro presidente do Brasil eleito pelo povo.

Você pode facilmente traduzir a primeira proposição para LQ, como

> *Pg*

À primeira vista, você poderia pensar em lidar com a segunda proposição com a mesma abordagem. Mas não pode, pois essas duas proposições, mesmo parecendo semelhantes, são na verdade muito diferentes.

A primeira proposição descreve Prudente de Morais em termos de *propriedade*, que provavelmente é compartilhada por outros (por exemplo, por Lula). É por isso que você pode usar facilmente uma *constante de predicado* para traduzir essa proposição para a LQ (veja o Capítulo 15 para saber mais sobre constantes de predicado).

A segunda proposição, porém, descreve Prudente de Morais em termos de *identidade* como o único primeiro presidente do Brasil eleito pelo povo. Para traduzir essa proposição para a LQ, você precisa de algo novo – uma forma de expressar identidade. Discuto os detalhes das identidades nos tópicos seguintes.

Discurso indireto e identidade

Existem algumas exceções interessantes à ideia de que *identidade* significa que podemos substituir livremente uma proposição por outra sem alterar seu sentido. Por exemplo, observe estas duas proposições:

Sara achava que Lula tinha sido o primeiro presidente do Brasil eleito pelo povo.

Sara achava que Lula era Prudente de Morais.

Obviamente, essas proposições não significam a mesma coisa; a primeira delas pode ser verdadeira e a segunda falsa.

Por esse motivo, a LQ não funciona com proposições que contêm *discurso indireto*, como

Élton *confiou* que Maria tivesse pagado a conta de gás.

O chefe *exigiu* que todos chegassem no horário.

Sabemos que Clarissa está mentindo.

É *necessário* que o sol nasça todas as manhãs.

Veja o Capítulo 21 para saber mais sobre os sistemas de Lógica não clássicos que tentam cuidar de proposições desse tipo.

Mas, lembre-se de que com relação à LQ, uma vez que você define uma identidade entre duas constantes, está dizendo que não há problema em substituir uma pela outra.

Entendendo as identidades

LEMBRE-SE

Uma *identidade* lhe diz que duas constantes individuais diferentes se referem à mesma coisa, significando que elas são intercambiáveis na LQ.

O que significa dizer que Prudente de Morais foi *o* primeiro presidente do Brasil? Na essência, você está dizendo que em qualquer lugar que falar sobre *Prudente de Morais*, pode substituir as palavras *o primeiro presidente do Brasil* e vice-versa.

Na LQ você pode afirmar uma identidade formalmente do seguinte modo (presumindo que *g* substitua *Prudente de Morais* e *f* o *primeiro presidente do Brasil*):

$g = f$

Você também pode usar a identidade com um quantificador:

$\exists x\ [x = f]$

Essa proposição é traduzida como: "existe um *x* tal que *x* é o primeiro presidente do Brasil" ou simplesmente "o primeiro presidente do Brasil existe".

Escrevendo provas com identidades

A LQ contém duas regras apenas para cuidar das identidades. Elas são tão fáceis de entender e usar que basta mostrar um exemplo de cada para que você as compreenda.

Regra da Identidade (ID)

A **ID** somente permite a substituição de uma constante por outra em uma prova depois que a identidade dessas constantes tiver sido demonstrada.

Depois que estabelece a identidade da forma $x = y$ em uma prova, a **ID** permite que você reescreva qualquer linha da prova, substituindo *x* por *y* (ou *y* por *x*)

Veja um argumento que requer uma identidade:

Premissas:

Toda pessoa merece respeito.

Jô Soares é uma pessoa.

Jô Soares foi o primeiro apresentador do *Programa do Jô.*

Conclusão:

O primeiro apresentador do *Programa do Jô* merece respeito.

Observe o mesmo argumento traduzido para a LQ:

$\forall x\ [Px \rightarrow Dx],\ Ps,\ s = o : Do$

A prova desse argumento é bem direta:

1.	$\forall x\ [Px \rightarrow Dx]$	**P**
2.	Ps	**P**
3.	$s = o$	**P**
4.	$Ps \rightarrow Ds$	1 **IU**
5.	Ds	2, 4 **MP**
6.	Do	5 **ID**

Usei a **ID** no último passo para substituir *o* por *s.* Na maioria dos casos, sempre que precisar usar a **ID** ela será bem óbvia.

Reflexividade da Identidade (RI)

A **RI** ainda é mais fácil de entender do que a **ID**. Ela diz que é possível presumir que tudo é idêntico a si mesmo, não importa o que você esteja provando.

A **RI** lhe permite inserir a proposição $\forall x\ [x = x]$ em qualquer ponto da prova.

Essa proposição $\forall x\ [x = x]$ é traduzida para o português como “qualquer que seja *x*, *x* é idêntico a *x*”, ou simplesmente “tudo é idêntico a si mesmo”.

A **RI** é uma daquelas regras de que você quase nunca precisa, a menos que o professor construa uma prova em um exame para que você tenha que usá-la (o que é sempre possível). Por exemplo, observe a prova seguinte:

$$\forall x\ [((x = m) \vee (x = n)) \rightarrow Tx] : Tm \wedge Tn$$

1.	$\forall x\ [((x = m) \vee (x = n)) \rightarrow Tx]$	**P**

2. $((m = m) \lor (m = n)) \to Tm$ 1 **IU**

3. $((n = m) \lor (n = n)) \to Tn$ 1 **IU**

Usei a **IU** para desenvolver a premissa de duas maneiras diferentes: na linha 2 trocando a variável *x* pela constante *m*, e na linha 3 trocando *x* por *n*. Agora você pode aplicar a **RI**:

4. $\forall x\ [x = x]$ **RI**

A **RI** funciona como uma premissa extra, de modo que você não precisa fazer referência ao número da linha quando a utiliza. Depois que usa a **RI**, você passa para a **IU** para obter as proposições de identidade de que precisa:

5. $m = m$ 4 **IU**

6. $n = n$ 4 **IU**

Dessa vez usei a **IU** para desenvolver a linha 4 de duas diferentes maneiras, mais uma vez troquei primeiro *x* por *m* e depois por *n.*

Agora tudo está em seu devido lugar e você pode concluir a prova usando somente as regras de inferência da LS:

7. $(m = m) \lor (m = n)$ 5 **Add**

8. $(n = m) \lor (n =n)$ 6 **Add**

9. Tm 2, 7 **MP**

10. Tn 3, 8 **MP**

11. $Tm \land Tn$ 9, 10 **Conj**

Capítulo 19

Plantando uma Quantidade de Árvores

Neste Capítulo

- Ampliando o método das árvores lógicas para as proposições da LQ
- Entendendo as árvores lógicas infinitas

No Capítulo 8, mostrei como usar as árvores lógicas na Lógica Sentencial (LS) para diversas finalidades. Neste capítulo, você verá como esse método pode ser estendido para a Lógica Quantitativa (LQ). As árvores lógicas na LQ geralmente são mais simples do que as provas, da mesma forma que ocorre na LS. Você não precisa de uma ideia genial para fazê-las funcionar, basta usar o velho método de "preencher e interpretar". Todavia, aviso desde já que as árvores lógicas da LQ têm limitações (infelizmente).

Neste capítulo, mostro como aproveitar ao máximo o método da árvore lógica para resolver problemas da LQ. Você verá ainda uma importante desvantagem — a árvore infinita.

Aplicando Seus Conhecimentos de Árvore Lógica na LQ

Tudo aquilo que você sabe sobre a construção de árvores lógicas na LS também se aplica na LQ. Neste tópico, lhe dou um exemplo de LQ para que aprenda como funciona. No caso de você se deparar com algo que não seja familiar e ficar emperrado, revise o Capítulo 8.

Usando as regras de decomposição da LS

Suponha que você quer testar a consistência do seguinte conjunto de três proposições:

$Ma \vee \sim Tb$

$\sim Ma \wedge Lc$

$\sim Tb$

Como nas árvores da LS, o primeiro passo para decidir se uma proposição é consistente ou inconsistente é construir o tronco usando as três proposições:

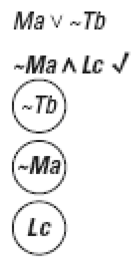
$Ma \vee \sim Tb$

$\sim Ma \wedge Lc$

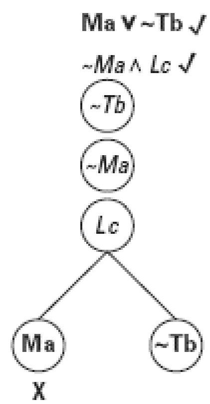
$\sim Tb$

Observe que circulei a terceira proposição porque ela já está em um formato que não pode mais ser desmembrado.

Agora posso construir a árvore usando as regras de decomposição da LS. Começarei com a segunda proposição, pois é de tronco único:



Depois de decompor a proposição *~Ma ∧ Lc*, revisei-a, inserindo o marcador (٧) para verificar o que havia feito com ela. Circulei, mais uma vez, as proposições resultantes que já foram totalmente decompostas. A única que ainda não está circulada ou assinalada é a primeira:



Agora todas as proposições estão assinaladas (٧) ou circuladas, o que significa que a árvore está terminada.

LEMBRE-SE

Observe que isolei o ramo terminado com *Ma* inserindo um X sob ele, assim como nas árvores da LS. Isso porque se você seguir o caminho do início do tronco até o final do ramo, passará por *~Ma* e *Ma*. Já que uma proposição e sua negação não podem ser ambas verdadeiras, o ramo deve ser isolado. Essa é a única razão para se isolar um ramo.

Mas o ramo que termina com *~Tb* não possui essa contradição e, assim, está aberto. Como ocorre com as árvores da LS, já que essa tem pelo menos um ramo aberto, o conjunto das três proposições é considerado *consistente*.

Adicionando IU, IE e NQ

LEMBRE-SE

Para as proposições com quantificadores, você precisa acrescentar as regras de quantificação do Capítulo 17. Para decompor as proposições em árvores, use a instanciação universal (**IU**) e a existencial (**IE**). Para remover o operador ~ do quantificador, use a negação do quantificador (**NQ**).

Observe este argumento com uma premissa que testei para verificar a validade usando a árvore lógica:

$\sim\exists x\ [Gx \wedge \sim Nx] : \forall x\ [Gx \rightarrow Nx]$

Como sempre, o primeiro passo é construir o tronco da árvore usando a premissa e a negação da conclusão:

$$\sim\exists x\ [Gx \wedge \sim Nx]$$

$$\sim\forall x\ [Gx \rightarrow Nx]$$

Já que ambas as proposições têm quantificadores negativos, use a **NQ** para mover os operadores ~ (observe que com a árvore você não tem que usar justificação nas linhas, tais como **NQ**, **IU** ou **IE**, como faz em uma prova). Economizei espaço dando dois passos de uma vez:

$$\sim\exists x\ [Gx \wedge \sim Nx]\ \checkmark$$
$$\sim\forall x\ [Gx \rightarrow Nx]\ \checkmark$$
$$\forall x \sim[Gx \wedge \sim Nx]$$
$$\exists x \sim[Gx \rightarrow Nx]$$

Agora você está pronto para usar a **IU** e a **IE**. Entretanto, lembre-se da limitação da **IE**: assim como nas provas da LS, você apenas pode usar a **IE** para liberar uma variável que ainda não esteja livre. Assim, neste exemplo, tem que tirar a **IE** do caminho, antes de usar a **IU**:

$\sim\exists x\,[Gx \wedge \sim Nx]\ \checkmark$

$\sim\forall x\,[Gx \to Nx]\ \checkmark$

$\forall x \sim[Gx \wedge \sim Nx]$

$\exists x \sim[Gx \to Nx]\ \checkmark$

$\sim[Gx \to Nx]$

$\sim[Gx \wedge \sim Nx]$

Quando usar a **IU,** não assinale a proposição que acabou de decompor. Mais tarde você saberá o porquê. Mas por ora apenas lembre-se da regra: quando usar a **IE**, assinale a proposição que foi decomposta, como sempre, mas não faça isso quando usar a **IU.**

Neste ponto, você pode começar a decompor as proposições usando as regras das árvores lógicas da LS. Quando possível, comece pela proposição que tenha um único ramo:

$\sim\exists x\,[Gx \wedge \sim Nx]\ \checkmark$

$\sim\forall x\,[Gx \to Nx]\ \checkmark$

$\forall x \sim[Gx \wedge \sim Nx]$

$\exists x \sim[Gx \to Nx]\ \checkmark$

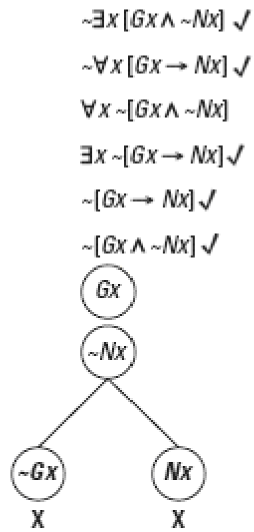
$\sim[Gx \to Nx]\ \checkmark$

$\sim[Gx \wedge \sim Nx]$

Gx

$\sim Nx$

Finalmente, decomponha a proposição de ramo duplo:



Ambos os ramos levam a contradições e, assim, os dois são isolados com um X. Tendo em vista que não há ramos abertos, a árvore está terminada e o argumento é considerado válido.

Usando a IU mais de uma vez

No exemplo do tópico anterior mostrei que quando você usa a **IU** para decompor uma proposição *não* deve assinalá-la como terminada. Prometi uma explicação e aqui vai ela.

Quando decompõe uma proposição ∀ usando a **IU** você tem um número ilimitado de constantes para escolher para a decomposição. Dessa forma, deixe a proposição não assinalada porque você poderá precisar usá-la mais uma vez.

Por exemplo, considere este argumento:

$$\forall x\ [Hx \rightarrow Jx],\ Ha \wedge Hb : Ja \wedge Jb$$

Monte o tronco da árvore, como de costume:

$\forall x\ [Hx \rightarrow Jx]$
$Ha \wedge Hb$
$\sim(Ja \,\square\, Jb)$

Sei que preciso decompor a primeira premissa usando a **IU**, mas tenho muitas opções. Poderia decompô-la para $Hx \rightarrow Jx$ ou $Ha \rightarrow Ja$ ou $Hb \rightarrow Jb$ ou um infinito número de outras proposições possíveis. Como você verá nesse exemplo, posso precisar de mais de uma decomposição para completar a árvore, de forma que tenho que deixar a proposição ∀ não assinalada para que possa decompô-la novamente quando necessário.

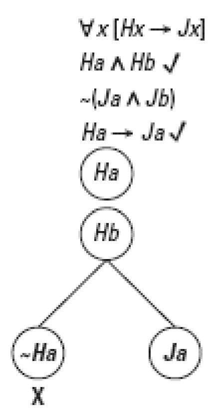
Na primeira decomposição, vou na direção que parece mais útil e, assim, escolhi uma decomposição que inclui uma das constantes que aparecem em outras proposições da árvore. Começo com a constante *a*:

$$\begin{array}{l} \forall x\,[Hx \rightarrow Jx] \\ Ha \wedge Hb \\ \sim(Ja \wedge Jb) \\ Ha \rightarrow Ja \end{array}$$

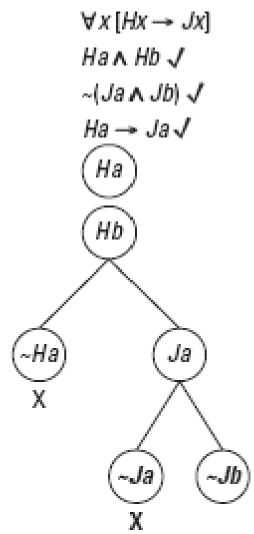
Deixei a proposição que acabei de decompor *não assinalada* e agora sigo para o próximo passo:

$$\begin{array}{l} \forall x\,[Hx \rightarrow Jx] \\ Ha \wedge Hb \ \checkmark \\ \sim(Ja \wedge Jb) \\ Ha \rightarrow Ja \\ Ha \\ Hb \end{array}$$

Desta vez, assinalei a proposição como de costume. Agora, tenho um ramo duplo:



Felizmente, um dos ramos tem que ser isolado, e sigo para o próximo passo:

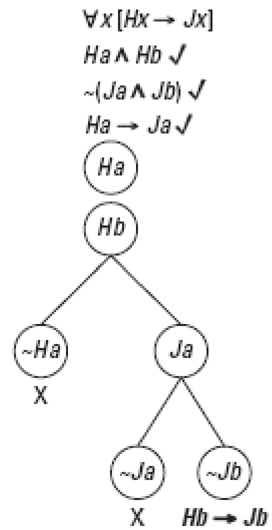


Você pode ficar tentado, nesse ponto, a achar que a árvore está terminada. Mas, lembre-se de que uma árvore só estará terminada quando:

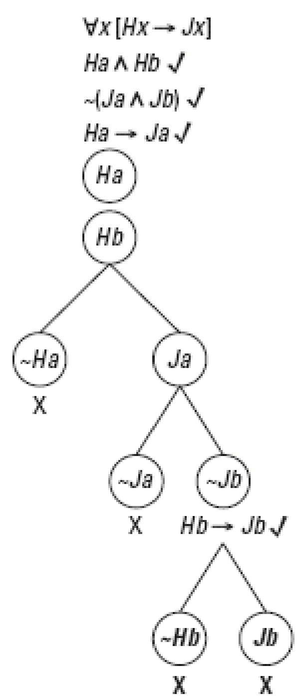
- ✓ Todos os itens tiverem sido assinalados ou circulados.
- ✓ Todos os ramos tiverem sido fechados.

Se essas regras não lhe soam familiares, veja o Capítulo 8. Agora vem o principal: a razão pela qual a árvore ainda não está terminada é porque quando usei a **IU** para decompor a primeira proposição, não a assinalei como terminada. Então, agora decomponho essa proposição de forma diferente, o que me permite completar a árvore.

Nesse caso, decomponho a proposição ∀ usando outra constante que aparece na árvore, ou seja, a constante *b*:



Observe que mesmo quando uso a **IU** para decompor a primeira proposição pela segunda (terceira ou centésima) vez, ainda a deixo *não assinalada*, caso precise usá-la novamente. Agora, basta apenas uma decomposição final e acabo:



Com essa última decomposição, todos os ramos estão isolados, o que significa que acabei e que o argumento é válido.

Árvores Infinitas

As árvores da LS são minha ferramenta de Lógica favorita, pois elas não requerem muita habilidade. Tenho apenas que seguir os passos corretos até o final, e todas as vezes terei a resposta certa.

Infelizmente, as árvores na LQ tendem a ser um pouco mais rebeldes. Em alguns casos, uma árvore cresce e cresce, e nunca para de crescer. Esse tipo é chamado de *árvores infinitas* (ou *intermináveis*), e elas tornam as coisas interessantes, para não dizer frustrantes.

Como você já deve saber, só há duas maneiras para terminar uma árvore: assinalando ou circulando todas as proposições *ou* fechando todos os ramos com um X. Mas na LQ nem sempre é possível chegar ao final de uma árvore.

Para ilustrar essa ideia, observe uma proposição traduzida para a LQ e testada usando a árvore lógica:

> Os números cardinais não são infinitos.

Essa é uma proposição *falsa*, pois os números naturais positivos (1, 2, 3...) continuam infinitamente, então, são infinitos. Mas essa é uma proposição que você pode expressar na LQ. Primeiro, declare o domínio de discurso:

> Domínio: Números naturais positivos

Agora defina a relação:

> Lxy = x é menor que y

Quero expressar a ideia de que não importa qual número eu escolha para x, esse número será sempre menor do que algum outro, denominado y que, então, negará tudo isso. Veja como escrever isso:

$\sim\forall x \, \exists y \, [Lxy]$

Isso é traduzido, literalmente, como “não é verdade que para todo *x*, existe um *y* tal que *x* é menor que *y”*, ou, simplesmente, “os números cardinais não são infinitos”.

Suponha que eu queira testar se essa proposição é uma tautologia. É melhor que *não* seja, pois já tenho uma interpretação que a torna falsa.

Como mostrei no Capítulo 8, para usar a árvore lógica para verificar se uma proposição é uma tautologia, você deve negá-la e usá-la como tronco. Quando completar a árvore, se houver pelo menos um ramo aberto, a proposição é uma tautologia; caso contrário, será uma contradição ou uma contingência.

Então, o primeiro passo é negar a proposição e usá-la como tronco da minha árvore e, então, decompô-la usando a IU, lembrando de *não* assinalá-la. Para economizar espaço, darei dois passos de uma vez:

$$\forall x \, \exists y \, [Lxy]$$
$$\exists y \, [Lxy]$$

Depois, decomponho essa nova proposição, usando a IE:

$$\forall x \, \exists y \, [Lxy]$$
$$\exists y \, [Lxy] \ \checkmark$$
$$Lxz$$

Nesse caso, decompus a proposição trocando a variável para *z* na esperança de que pudesse ser capaz de me livrar dela mais tarde aplicando a IU novamente na primeira proposição.

Nesse ponto, a árvore não está terminada, pois um ramo está aberto e a proposição não está assinalada. Mas ainda há uma chance de poder fechar o ramo usando a IU de maneira engenhosa na primeira proposição. Já que a variável z está em jogo, uso esta constante na decomposição:

$$\begin{array}{l} \forall x\, \exists y\, [Lxy] \\ \exists y\, [Lxy] \checkmark \\ Lxz \\ \exists y\, [Lzy] \end{array}$$

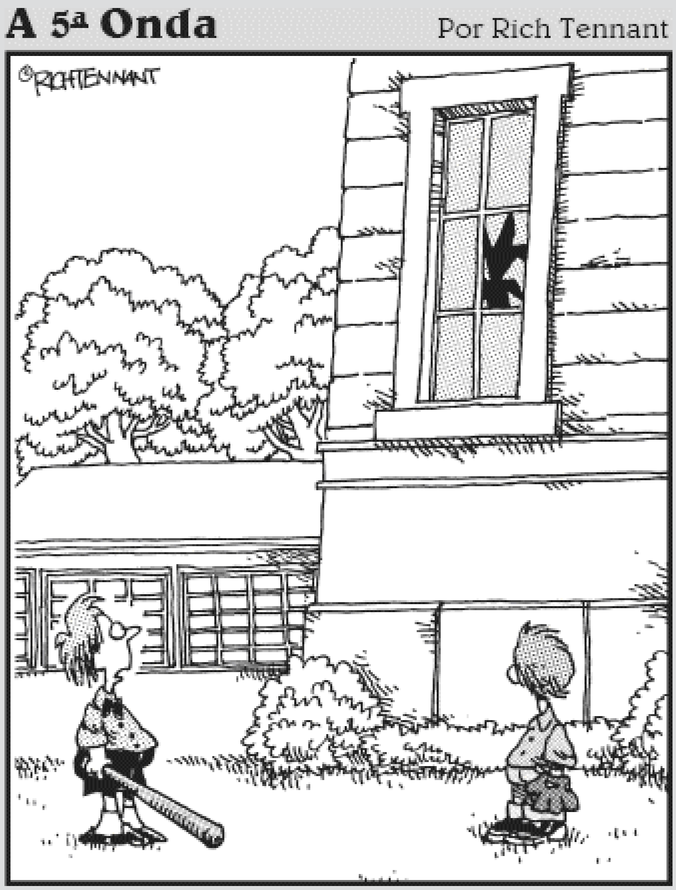
Agora, outra aplicação da IE:

$$\begin{array}{l} \forall x\, \exists y\, [Lxy] \\ \exists y\, [Lxy] \checkmark \\ Lxz \\ \exists y\, [Lzy] \checkmark \\ Lzw \end{array}$$

Dessa vez, inseri a variável *w*, mas isso não importa. Na verdade, não há nada que você possa fazer para completar essa árvore, seja fechando um ramo ou assinalando a primeira proposição, pois o que temos aqui é uma árvore infinita.

Já que a árvore nunca tem fim, ela é incapaz de dizer se a proposição que você está testando é uma tautologia. As árvores infinitas são o motivo pelo qual as árvores lógicas têm uma função mais limitada na LQ, apesar de sempre serem úteis na LS.

Parte V
Desenvolvimentos Modernos na Lógica



"É um argumento válido, mas ainda precisamos tirar aquela bola de lá."

Nesta parte...

A Lógica pode ter começado com Aristóteles, mas certamente não terminou com ele. A Parte V o atualizará, discutindo a Lógica a partir do século XX.

No Capítulo 20, você descobrirá como a Lógica é um instrumento na computação, tanto no hardware quanto no software. O Capítulo 21 apresenta alguns exemplos da Lógica Não Clássica, que são as formas que partem de conjuntos diferentes de suposições daquelas discutidas no restante do livro. Mostrarei, ainda, algumas diferenças surpreendentes entre o que parece óbvio e o que é possível na Lógica. Finalmente, o Capítulo 22 analisa como os paradoxos desafiam a Lógica e como as questões de consistência e completude conduziram a mais importante descoberta matemática do século.

Capítulo 20
Lógica Computacional

Neste Capítulo

- Avaliando os primeiros computadores
- Entendendo como a Lógica funciona nos computadores atuais

O computador tem sido chamado de a mais importante invenção do século XX (exceto, talvez, pela cafeteira automática). Ele se distingue de todas as outras invenções — por exemplo, o avião, o rádio, a televisão ou o gerador de energia nuclear — por sua versatilidade.

Quando você pensa sobre isso, conclui que as máquinas, em sua maioria, são apenas ferramentas para realizar trabalhos repetitivos que os humanos não gostam de fazer, e, sinceramente, máquinas em geral fazem esse trabalho muito melhor do que os humanos jamais poderiam fazer. Do abridor de latas ao lava rápido, elas têm sido construídas há muito tempo para imitar e aperfeiçoar o movimento humano.

Assim sendo, faz todo sentido que durante esse processo as pessoas tenham começado a imaginar se poderiam construir máquinas que assumissem alguns dos trabalhos *mentais* repetitivos que os humanos têm que fazer diariamente. Embora máquinas de somar e caixas registradoras, por exemplo, tenham sido inventadas para fazer apenas isso, essas invenções são limitadas em sua capacidade de realizar coisas. Assim, não se pode esperar que um abridor de latas lave um carro nem que uma máquina de somar faça divisões longas ou, muito menos, cálculo diferencial.

Alguns visionários, porém, pensaram na possibilidade de uma única máquina ser capaz de realizar um ilimitado número de funções.

Neste capítulo, mostro o papel da Lógica na idealização dos computadores. Começo por seu surgimento, com o trabalho de Charles Babbage e Ada Lovelace. Em seguida, discuto como Alan Turing demonstrou, ao menos na teoria, que um computador poderia realizar qualquer tipo de cálculo que uma pessoa pudesse. Finalmente, enfoco nas maneiras com que a Lógica forma a base do computador, tanto no nível do hardware quanto do software.

As Primeiras Versões dos Computadores

Ainda que o trabalho para a construção do primeiro computador eletrônico tenha começado no anos 1940, a ideia e o projeto surgiram mais de um século antes. O computador teve início como uma ideia maluca que nunca poderia ser concretizada e se transformou em uma das mais importantes invenções da história.

Babbage projeta os primeiros computadores

Charles Babbage (1791-1871) é considerado o inventor do computador. Ainda que os dois modelos criados por ele — a máquina diferencial e, mais tarde, o engenho analítico — tenham sido desenvolvidos para serem alimentados por energia mecânica e não elétrica, eram máquinas altamente sofisticadas e tinham muito mais em comum com os computadores mais recentes do que as outras invenções da época.

Babbage começou seu trabalho na máquina diferencial nos anos 1820. Ainda que tenha concluído o projeto, infelizmente nunca conseguiu construir a máquina. As dificuldades de financiamento e os conflitos de personalidade entre Babbage e as outras pessoas envolvidas no projeto foram considerados os motivos do seu fracasso (mas, tenho certeza de que mesmo os engenheiros de computação atuais enfrentam esses tipos de obstáculos).

A primeira e única máquina diferencial foi construída somente em 1991, seguindo os planos de Babbage. Os construtores se limitaram à tecnologia disponível na sua época. Eles descobriram que a máquina funcionava como havia sido planejada, realizando cálculos matemáticos complexos com perfeita exatidão.

Depois que Babbage abandonou seus planos de construir a máquina diferencial iniciou um projeto ainda mais ambicioso — desenhar o engenho analítico. Esse projeto incorporou as habilidades que ele já havia adquirido com a máquina diferencial, mas ele deu um passo adiante. O grande aperfeiçoamento do engenho analítico foi sua possibilidade de programação através de cartões perfurados, o que o tornava ainda mais versátil e fácil de usar do que a máquina diferencial. Ada Lovelace, matemática e amiga de Babbage, foi de grande ajuda no projeto do engenho analítico. Ela ainda elaborou diversos programas que seriam executados no engenho se ele tivesse sido construído.

Turing e sua MUT

Depois da morte de Charles Babbage, em 1871, seus projetos foram abandonados por décadas, até que outro visionário, Alan Turing (1912-1954), abordasse a ideia da computação mecanizada de um modo diferente.

Turing vislumbrou a necessidade de elucidar exatamente o que significava computação, que já era realizada há séculos, usando *algorítimos*, que são simples procedimentos mecânicos que produzem o resultado desejado em um número limitado de passos. Por exemplo, o procedimento para a multiplicação de dois números é um algorítimo. Se você seguir os passos corretamente, a resposta certa estará garantida, não importa o tamanho dos números.

Turing descobriu que os algorítimos poderiam ser desmembrados em passos menores e como fazer isso através de um aparelho extremamente simples, que foi chamado de Máquina Universal de Turing (MUT).

Ao contrário de Babbage, Turing nunca esperou que sua máquina fosse construída. Ela foi usada como modelo teórico e poderia ser descrita como princípios, sem ter sido implementada. Entretanto, a capacidade básica de uma Máquina Universal de Turing é compartilhada por todos os computadores. Em outras palavras, todo computador, não importa qual seja seu projeto, não é mais nem menos capaz de realizar cálculos do que qualquer outro.

Explicando a MUT

A MUT consiste de uma tira de papel de extensão determinável dividida em quadrados. Cada quadrado contém um único símbolo de um conjunto finito de símbolos. A tira de papel é presa em rolos para que se mova, um quadrado por vez, diante de um cabeçote que lê o símbolo contido naquele quadrado e, em alguns casos, apaga-o para escrever um símbolo diferente.

Por exemplo, para elaborar um programa para multiplicar dois números, você começaria escrevendo esses números em uma tira, separando-os com o operador de multiplicação.

```
                   v
| | | | | | | | | |7|5|8|X|6|3| | | | | | | | | | | | |
```

Esses números seriam as *condições iniciais* do programa. Ao final do processo, o resultado ficaria assim:

```
                   v
| | | | | | | | | |7 5|8|X|6|3|=|4|7|7|5|4| | | | | | |
```

LEMBRE-SE

Turing traçou um conjunto de passos possíveis do começo ao fim. Eles formam o que chamamos de *programa*, que consiste de uma lista de *estados* que dizem à máquina as *ações* a serem tomadas com base naquilo que o cabeçote lê. Dependendo do estado em que a máquina esteja, ela realiza ações específicas e diferentes uma da outra. Essas ações são, de maneira geral, as seguintes:

1. Optar sobre a alteração de um símbolo por outro no quadrado.
2. Optar sobre a alteração do estado em que a máquina se encontra.
3. Mover um quadrado para a direita ou para a esquerda.

Por exemplo, quando a máquina está no estado nº 1 e o cabeçote lê o número 7, as ações podem ser

1. Deixar o número 7 inalterado.
2. Mudar a máquina do estado nº 1 para o nº 10.
3. Mover um quadrado para a direita.

No entanto, quando a máquina está no estado nº 2 e o cabeçote lê o número 7, as ações podem ser

1. Mudar o número 7 para 8.
2. Deixar o estado inalterado.
3. Mover um quadrado para a esquerda.

Ainda que essas ações sejam rudimentares, Turing demonstrou que é possível realizar cálculos sofisticados dessa maneira. Mais importante ainda, provou que é possível realizar *qualquer* cálculo dessa forma. Isto é, a MUT poderia ser programada para realizar qualquer método algorítimico que uma pessoa pudesse aprender a fazer.

Relacionando a MUT e a Lógica

Mas você pode estar se perguntando: o que a MUT tem a ver com a Lógica? Para entender a conexão entre elas, observe as especificações que fazem com que a máquina decida o valor verdade de certos tipos de proposições simples. No exemplo da multiplicação que utilizei anteriormente, a seguinte proposição é verdadeira:

> A máquina está no estado nº 1 e o cabeçote está no 7.

Considerando que a proposição é verdadeira, a máquina realiza as ações adequadas, o que a reinicia de forma que agora esta proposição seja verdadeira:

> A máquina está no estado nº 10 e o cabeçote está no 5.

A Lógica é ideal para descrever as condições da máquina em qualquer momento. Como você verá no

LEMBRE-SE

tópico a seguir, os cientistas da computação usaram o poder descritivo da Lógica no projeto tanto do hardware quanto do software.

A Era Moderna dos Computadores

As ideias e teorias iniciais de Babbage e Turing, pioneiros no desenvolvimento dos computadores, prepararam o terreno para o desenvolvimento do computador moderno. Com a energia elétrica se tornando mais e mais comum, o avanço seguinte na computação veio logo após. Construído em 1940, o ENIAC (sigla em inglês para Computador e Integrador Numérico Eletrônico) foi o primeiro computador eletrônico e originou modelos mais aperfeiçoados, que foram sendo desenvolvidos rapidamente.

Todos os computadores já desenvolvidos (até agora), incluindo seu computador pessoal ou laptop, podem ser divididos em dois níveis de funcionalidade.

> ✓ **Hardware:** a estrutura física do computador
>
> ✓ **Software:** os programas que são executados nele

Como já explicado neste tópico, a Lógica é parte integrante da computação, do software e do hardware.

Hardware e portas lógicas

A *função lógica* pega um ou dois valores de entrada e gera um valor de saída. Os valores de entrada e de saída podem ser um dos valores lógicos **V** ou **F**. (veja o Capítulo 13 para uma revisão sobre as funções verdadeiro).

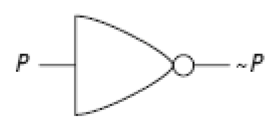
Todos os cinco operadores da Lógica Sentencial (LS) são funções lógicas. Três desses operadores (~, ^ e ˅) são suficientes para elaborar todas as funções lógicas possíveis (veja o Capítulo 13). Da mesma forma, o operador |, (também conhecido como traço de Sheffer) é suficiente para elaborar, sozinho, todas as funções verdadeiro possíveis.

No nível mais básico, circuitos de computador imitam as funções verdadeiro. Porém, ao invés de usar os valores **V** e **F**, o computador usa os valores 1 e 0, como na Álgebra Booleana (veja o Capítulo 14). Em homenagem a George Boole, as variáveis que usam esses dois valores são chamadas de *variáveis booleanas*. Quando a corrente está ligada em uma parte específica do circuito, o valor nela é considerado 1; quando está desligada, o valor é 0.

Tipos de portas lógicas

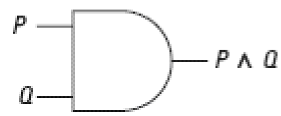
O comportamento dos operadores é mimetizado pelas *portas lógicas*, que permitem que a corrente passe através dos circuitos em um modo predeterminado. As seis portas mais comuns são NOT (NÃO), OR (OU), AND(E), NAND (NÃO e E), NOR (NÃO e OU) e XOR (OU EXCLUSIVO).

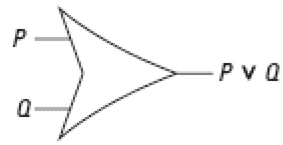
Por exemplo, observe o diagrama para a porta NOT:



Em termos booleanos, uma porta NOT altera uma entrada de 0 para uma saída 1 e uma entrada 1 para uma saída 0. Em outras palavras, quando a corrente de entrada está ligada, a corrente de saída está desligada; quando a corrente de entrada está desligada, a corrente de saída está ligada.

A porta NOT é a única que possui apenas uma entrada. Ela se assemelha ao operador ~, que é o único operador unário da LS (por *unário* quero dizer que o operador ~ é colocado na frente de *uma* constante e não entre *duas* delas). As demais portas têm duas entradas. Por exemplo, observe o diagrama para as portas AND e OU, respectivamente:





A porta AND tem corrente de saída somente quando ambas as suas fontes de corrente de entrada estão ligadas; caso contrário, a corrente de saída permanece desligada. Como você já deve ter adivinhado, uma porta OR tem corrente de saída quando qualquer de suas fontes de entrada está ligada; caso contrário, a saída permanece desligada.

Ainda que as portas NOT, AND e OR sejam suficientes para imitar qualquer função verdadeiro da LS, é aconselhável, por questões práticas, que se tenha uma maior variedade de matéria-prima. As portas a seguir são as mais utilizadas:

> ✓ **NAND:** abreviação das palavras *não* e *e, em* inglês. Esta porta funciona de modo semelhante ao operador |, (veja o Capítulo 13). A saída é 0 somente quando ambas as suas entradas são 1; caso contrário, sua saída é 0.
>
> ✓ **XOR:** abreviação em inglês de *ou exclusivo*. Esta porta funciona como a função “ou exclusivo”, que você viu no Capítulo 13. Isto é, quando uma entrada específica é 1, a saída também é 1; caso contrário, a saída é 0.
>
> ✓ **NOR:** abreviação de *não* e *ou*, em inglês. Essa porta tem corrente de saída somente quando ambas as suas entradas estão desligadas; caso contrário, a corrente de saída permanece desligada.

Computador e portas lógicas

As portas são a matéria-prima da CPU (unidade de processamento de dados) do computador, que é o local deste onde os dados são manipulados. Esses dados são armazenados em dispositivos de memória e transportados para dentro e para fora da CPU, quando necessário, mas o real “raciocínio” que ocorre no computador acontece na CPU.

Como o trabalho de Turing demonstrou (veja o tópico anterior “Turing e sua MUT”), uma breve lista de manipulações de dados é suficiente para calcular tudo o que pode ser calculado. As portas lógicas fornecem muito mais do que as funcionalidades básicas para reproduzir uma Máquina Universal de Turing.

Acredite ou não, o engenho analítico de Charles Babbage, já estudado neste capítulo, que executa programas em cartões perfurados, não teria nem mais nem menos capacidade computacional do que os computadores mais modernos de hoje. É claro que os computadores de hoje têm toneladas a mais de memória e realizam todos esses processos na velocidade da luz comparados ao modelo de Babbage. Mas, a princípio, ambas as máquinas são suficientes para realizar todos os cálculos matemáticos possíveis.

Software e linguagem de computador

O hardware do computador por si só é muito mais complexo do que qualquer outra máquina que você possa imaginar. Ainda assim, não importa o quanto esse mecanismo básico do computador seja complexo, ele está limitado a realizar as funções para as quais foi designado — como qualquer outra máquina.

Essas capacidades limitadas são tudo o que é necessário para algumas aplicações. Os circuitos computacionais de carros, relógios e utensílios domésticos, por exemplo, realizam um excelente trabalho no controle das máquinas onde estão instalados. Mas se você tentar usar os circuitos de uma BMW para controlar sua máquina de lavar louças — ou mesmo um carro diferente — ficará desapontado com os resultados.

Então, por que o seu PC ou Mac pode realizar um número infinito de tarefas? A resposta, é claro, está no software.

Software é qualquer programa de computador que diz ao hardware como executar as tarefas. Todo software é escrito em uma das muitas *linguagens de programação*, tais como Java, *C* ++, Visual Basic, COBOL e Ada (batizada em homenagem a Ada Lovelace, de quem já falei neste capítulo). Ainda que todas as linguagens de computador tenham diferenças na sintaxe e em seus pontos fortes e fracos, todas têm uma coisa em comum: a Lógica.

Como na LS, a linguagem de computação lhe permite declarar as constantes e variáveis às quais você poderá atribuir valores. Por exemplo, se quiser declarar uma variável chamada “mês” e determinar seu valor inicial como sendo 10, você terá que escrever as seguintes linhas de código computacional em diferentes linguagens:

- ✓ **Java:** int mes = 10;
- ✓ **Visual Basic:** Dim mes as Integer = 10
- ✓ **PL/I:** DCL MES FIXED BINARY (31,0) INIT (10);

Como você pode ver, em cada linguagem a sintaxe é diferente, mas a ideia principal é a mesma. Depois que você tem as variáveis para trabalhar, pode construir proposições que testam se determinadas condições são verdadeiras ou falsas, e, então, pode agir com base nos resultados.

Por exemplo, se quiser enviar uma mensagem, se e somente se a data for 12 de junho, terá que escrever uma proposição *se* em cada uma das três linguagens para testar se o mês e o dia estão corretos e, se estiverem, como realizar a ação apropriada:

> ✓ **Java:**
>
> ```
> if (mes == 06 && dia == 12)
>    mensagem = “Feliz Dia dos Namorados!”;
> ```
>
> ✓ Visual Basic:
>
> ```
> if mes = 06 And Dia = 12 Then _
>    mensagem = “Feliz Dia dos Namorados!”
> ```
>
> ✓ PL/I:
>
> ```
> IF MES = 06 & DIA = 12 THEN
>    MENSAGEM = ‘Feliz Dia dos Namorados!’;
> ```

Novamente, a sintaxe muda de uma linguagem para outra, mas o significado é o mesmo.

Você pode ver como a estrutura da linguagem de programação permite que use o computador para fazer, de uma maneira mais eficiente, aquilo que uma Máquina Universal de Turing só conseguiria fazer em minúsculos passos. Contudo, a ideia básica por trás de ambos os tipos de computação é a mesma:

1. Estabeleça as condições iniciais.
2. Teste as condições vigentes e faça as alterações adequadas quando necessárias.
3. Repita o Passo 2 quantas vezes for necessário até concluir o processo.

Capítulo 21

Proposições Indecidíveis: A Lógica Não Clássica

Neste Capítulo

- Apresentando a possibilidade com a Lógica Multivalorada e Difusa
- Entendendo a Lógica Modal, de Ordem superior e Paraconsistente
- Avaliando o mistério da Lógica Quântica

Para a maioria das pessoas, a pergunta "e se 2 + 2 = 5?" é absurda. Quero dizer que se você ficar se perguntando "e se...?", na maioria das vezes não tem sentido, porque sabe que nunca poderia ser verdade. Por exemplo, veja esta improvável pergunta: "e se pequenos marcianos verdes pousassem seu disco voador em frente de minha casa e fugissem com meu carro?".

Mas para os estudiosos da Lógica, essas perguntas "E se?" são proposições indecidíveis — um desafio instigante justamente por serem tão irracionais. Até o início do século XX, a Lógica tinha sido reduzida a uma lista relativamente curta de suposições básicas chamadas de *axiomas.* Veja o Capítulo 22 para saber mais sobre os axiomas na Lógica. Eles eram considerados evidentes e quando você os aceitava como verdadeiros todo o resto se desenvolvia logicamente.

Mas e se você não os aceitasse? E se, para fins de argumentação, você modificasse um axioma da mesma maneira que um confeiteiro muda a receita de um bolo?

É claro que um confeiteiro deve ter cuidado ao trocar os ingredientes. Por exemplo, substituir fermento em pó por açúcar pode impedir que o bolo cresça e substituir chocolate por alho transformaria o bolo em algo que nem seu cachorro iria comer.

Da mesma forma, os lógicos precisam escolher cuidadosamente quando mudam os axiomas. Mesmo uma pequena alteração pode resultar em um sistema lógico cheio de contradições. Ainda que um sistema seja consistente, ele pode ser provado banal, desinteressante ou inútil — o bolo de alho da Lógica. Mas, dadas essas dificuldades, alguns poucos dissidentes ainda têm criado sistemas alternativos de Lógica.

Bem-vindo ao mundo da *Lógica Não Clássica*, a resposta moderna para mais de 2.000 anos de *Lógica Clássica* (que inclui quase tudo que é tratado neste livro). Neste capítulo, você terá uma amostra de diversos sistemas de Lógica Não Clássica. Esses sistemas não só desafiam suas noções de verdadeiro e falso como também vão surpreendê-lo em relação a quanto podem ser úteis no mundo real.

Abertos para as Possibilidades

No Capítulo 1, apresentei-o à *lei do terceiro excluído*, que é a regra pela qual uma proposição é verdadeira ou falsa, não sendo possível meio-termo. Essa regra é uma das importantes suposições da Lógica e, por milhares de anos, foi muito útil.

Quando expliquei essa regra, deixei claro que nem *tudo* no mundo se encaixa tão harmoniosamente; nem deve se encaixar. A proposição "Hugo é alto" pode ser verdadeira quando Hugo está entre um grupo de crianças e falsa quando ele está ao lado de jogadores de basquete.

Os estudiosos consideram essa objetividade uma das limitações da Lógica, que não aceita meios-termos. Se você quiser trabalhar com essa proposição, terá que estabelecer uma definição para a palavra *alto*. Por exemplo, pode estabelecer que um homem com um metro e oitenta de altura ou mais é *alto* e todos os demais *não são*.

A lei do terceiro excluído foi a pedra fundamental da Lógica de Aristóteles até o século XX. Essa lei é uma das coisas que simplesmente você aceita como uma suposição necessária se quiser chegar a algum lugar com a Lógica. Mas, em 1917, um homem chamado Jan Lukasiewicz começou a pensar sobre o que aconteceria se um terceiro valor fosse incorporado à Lógica. Veja nos próximos tópicos onde essas ponderações levaram Lukasiewicz — e a Lógica como um todo.

Lógica Trivalente

Jan Lukasiewicz decidiu analisar o que aconteceria se um terceiro valor, que não fosse nem verdadeiro nem falso, fosse adicionado à Lógica. Ele chamou esse terceiro valor de *possível*, e declarou que ele seria atribuído às proposições onde a veracidade fosse inconclusiva, tais como:

> Choverá em São Paulo amanhã.
>
> Os médicos algum dia encontrarão a cura para a gripe.
>
> Existe vida em outros planetas.

O possível parece estar entre o verdadeiro e o falso. Por essa razão, Lukasiewicz começou com uma representação Booleana de 1 para verdadeiro e 0 para falso. Veja o Capítulo 14 para saber mais sobre a Lógica Booleana. A partir daí ele adicionou um terceiro valor, ½, para o possível.

O resultado foi a *Lógica Trivalente*, que utiliza os mesmos operadores da Lógica Clássica, mas com novas definições para abranger esse novo valor. Por exemplo, observe a tabela verdade para ~*x:*

x	$\sim x$
1	0
½	½
0	1

Esse valor para a possibilidade faz sentido quando você a utiliza em um exemplo:

Considere *P* = Choverá em São Paulo amanhã.

Se o valor verdade de *P* é ½ — isto é, se isso é possível —, também é possível que não chova em São Paulo amanhã. Então o valor de ~*P* também é ½.

Pela mesma linha de raciocínio, você pode descobrir o valor verdade de proposições ∧ e ∨ que incluem uma ou mais subproposições cujos valores sejam possíveis. Por exemplo:

Ou Pedro Álvares Cabral descobriu o Brasil *ou* choverá amanhã em São Paulo.

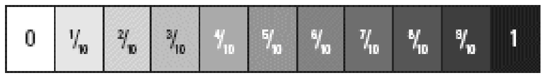
Essa proposição é verdadeira (a primeira parte é verdadeira, então o resto não importa), assim, seu valor é 1.

Lógica Multivalorada

Depois que a porta para esse terceiro valor verdade se

LEMBRE-SE

abre, você pode criar um sistema lógico com mais de um valor intermediário — na verdade, qualquer número que quiser. Por exemplo, imagine um sistema com 11 valores, de 0 a 1 com intervalos de 1/10. Esse sistema é um exemplo de *Lógica Multivalorada*.



Como você computa o valor verdade das proposições com esse sistema? Na Lógica Multivalorada as regras são simples:

- ✓ **Regra do ~:** ~*x* significa 1 – *x.*
- ✓ **Regra do ∧:** *x* ∧ *y* significa *escolha o valor menor.*
- ✓ **Regra do ∨:** *x* ∨ *y* significa *escolha o valor maior.*

Essas três regras podem parecer estranhas, mas funcionam. Você ainda pode usá-las com a Lógica Trivalente e com a Álgebra Booleana.

Por exemplo:

~2/10 = 8/10	**regra do ~**
3/10 ∧ 8/10 = 3/10	**regra do ∧**
1/10 ∨ 6/10 = 6/10	**regra do ∨**
7/10 ∧ 7/10 = 7/10	**regra do ∧**
9/10 ∨ 9/10 = 9/10	**regra do ∨**

DICA

Você pode até adicionar os outros dois operadores da Lógica Sentencial (LS) usando as regras da equivalência **Impl** e **Equiv**. (veja o Capítulo 10 para saber mais sobre as regras de equivalência):

- ✓ **Impl:** *x* → *y* = ~*x* ∨ *y*
- ✓ **Equiv:** *x* ↔ *y* = *(x* → *y)* ∧ (*y* → *x*)

Com essas regras da Lógica Multivalorada, você pode calcular o valor de qualquer expressão, passo a passo, do mesmo modo que uma prova. Por exemplo:

$4/10 \leftrightarrow 3/10$	
$= (4/10 \rightarrow 3/10) \wedge (3/10 \rightarrow 4/10)$	**Equiv**
$= ({\sim}4/10 \vee 3/10) \wedge ({\sim}3/10 \vee 4/10)$	**Impl**
$= (6/10 \vee 3/10) \wedge (7/10 \vee 4/10)$	**Regra do ~**
$= 6/10 \wedge 7/10$	**Regra do ∨**
$= 6/10$	**Regra do ∧**

Jan Lukasiewicz focou na sintaxe do sistema e deixou a semântica aberta a interpretações. Veja o Capítulo 14 para mais detalhes sobre sintaxe e semântica. Em outras palavras, o cálculo na Lógica Multivalorada é totalmente fundamentado em regras, mas o significado do resultado está aberto a interpretação. No tópico a seguir, “Lógica Difusa”, descrevo várias maneiras de interpretar os resultados.

Lógica Difusa

Alguns críticos da Lógica Multivalorada questionaram sua utilidade para descrever eventos futuros possíveis. Eles citam a Teoria da Probabilidade como a ferramenta mais apropriada para examinar o futuro (a teoria da probabilidade também calcula a possibilidade, mas usa um método de cálculo diferente).

Esses críticos têm seu mérito. Por exemplo, se a chance de chover em São Paulo é 3/10 e no Rio de Janeiro é 7/10, a teoria da probabilidade calcula a chance de chover nos *dois* lugares multiplicando esses valores:

$$3/10 \times 7/10 = 21/100$$

A Lógica Multivalorada, por sua vez, calcularia esse valor como sendo 3/10. Mas, como a teoria da probabilidade é muito bem estabelecida, essa discrepância entre os dois cálculos questiona a utilidade da Lógica Multivalorada.

Nos anos 1960, porém, um matemático chamado Lotfi Zadeh viu o potencial da Lógica Multivalorada para representar não aquilo que é possivelmente verdadeiro, mas sim o que é *parcialmente verdadeiro.*

Considere a seguinte proposição:

> Estou com fome.

Depois que você comer bastante, essa proposição será, provavelmente, falsa. Mas se ficar sem comer por várias horas, em algum momento ela será verdadeira novamente. Entretanto, a maioria das pessoas não percebe essa mudança como algo “preto ou branco”, mas sim em tons de cinza; isto é, acham que a proposição vai se tornando mais verdadeira (ou falsa) com o passar do tempo.

Na verdade, a maioria dos chamados opostos — alto ou baixo, quente ou frio, feliz ou triste, ingênuo ou sagaz e assim por diante — não são separados por linhas bem definidas. Ao invés disso, são extremos de uma linha contínua que contém discretas variações. Na maioria das vezes, graduações desse tipo são, de certo modo, meramente subjetivas.

A resposta de Zadeh a essas questões é a *Lógica Difusa*, que é uma extensão da Multivalorada e, assim como ela, permite a existência de valores como 0 para completamente falso, 1 para completamente verdadeiro e intermediários para descrever graduações da verdade. Mas, ao contrário da Lógica Multivalorada, todos os valores entre 0 e 1 são permitidos.

O preço da TV nova

Para entender a Lógica Difusa um pouco melhor, considere um casal hipotético, Dora e João. Eles concordaram em comprar uma TV nova para a casa deles. Dora está pensando em um modelo de tamanho médio, que custe cerca de $500, mas João tem em mente uma gigantesca tela de plasma, que custa cerca de $2.000. É claro que quando eles chegaram na loja perceberam que havia um problema. Se encarassem a controvérsia sob a perspectiva da Lógica Bivalente, eles nunca chegariam a uma decisão.

Depois de discutirem a questão, ficou claro que existia alguma flexibilidade nas opiniões de ambos.

Como na Lógica Multivalorada, as proposições $\wedge$ na Lógica Difusa são encaradas como sendo o *menor* de dois valores e as proposições $\vee$ como o *maior* deles. Nesse caso, a questão requer uma proposição $\wedge$: Dora *e* João devem chegar a um acordo quanto ao valor que pretendem gastar.

Uma pequena anotação será útil aqui:

Considere $D(x)$ = O valor lógico de Dora é x reais.

Considere $J(x)$ = O valor lógico de João é x reais.

Agora você pode construir esta proposição para compreender a reação combinada deles a um valor específico:

$$D(x) \wedge J(x)$$

Então, quando um vendedor lhes mostra as televisões na faixa de $500, ele obtém um 1 de Dora e um 0 de João:

$$D(500) \wedge J(500) = 1 \wedge 0 = 0$$

Porém, quando mostra as televisões na faixa de $2.000, ele obtém um 0 de Dora e um 1 de João:

$$D(2000) \wedge J(2000) = 0 \wedge 1 = 0$$

Quando o vendedor mostra as de preço intermediário, o impasse começa a se desfazer. Dora e João decidiram por uma TV de $1.100, que ambos acharam ser o maior valor possível:

$$D(1100) \wedge J(1100) = .75 \wedge .75 = .75$$

Comprando votos

É claro que nem todos os problemas podem ser decididos por uma proposição $\wedge$. Em alguns casos, o caminho é o da proposição $\vee$. Se for o caso, a Lógica Difusa ainda consegue resolver a questão.

Por exemplo, suponha que você seja um político que precisa de mais um voto na Câmara de Vereadores para aprovar um projeto. Os dois votantes que se abstiveram são Rita e Jorge, ambos estão esperando para saber quanto de dinheiro você alocará para a construção de um playground em uma escola antes de proferir seus votos. Rita espera que o valor seja de $5.000 e Jorge quer que seja de $25.000.

Já que só precisa de um voto, você usa uma proposição ∨, da seguinte forma:

$$R(x) \vee J(x)$$

A princípio você pensa que um acordo pode funcionar e, assim, tenta dividir a diferença, para agradar a todos, chegando a um valor de $15.000:

$$R(15000) \vee J(15000) = .1 \vee .1 = .1$$

Não é um bom resultado. Nesse cenário, ambos os vereadores ficarão tão descontentes que provavelmente votarão contra o seu projeto. Então você experimenta os dois valores, $5.000 e $25.000:

$$R(5000) \vee J(5000) = 1 \vee 0 = 1$$

$$R(25000) \vee J(25000) = 0 \vee 1 = 1$$

Quanta diferença! Você pode escolher qualquer desses valores e fazer com que pelo menos um dos vereadores fique satisfeito, que é tudo o que você precisa para garantir a aprovação.

Conhecendo uma Nova Modalidade

Assim como na Multivalorada, *a Lógica Modal* tenta lidar não apenas com o verdadeiro e o falso, mas também com o possível. Por essa razão a Lógica Modal traz dois novos operadores: o *de necessidade* e o *de possibilidade.*

$\Diamond x$ = É possível que *x.*

$\Box x$ = É necessário que *x.*

Por exemplo:

Considere *C* = Círculos são redondos.

Então, $\Diamond C$ significa "é possível que círculos sejam redondos", e $\Box C$ significa "é necessário que círculos sejam redondos".

Os dois operadores modais são logicamente relacionados da seguinte forma:

$\Box x = \sim\Diamond\sim x$

Por exemplo, a proposição "é necessário que círculos sejam redondos" é equivalente a "não é possível que círculos não sejam redondos". Da mesma forma, a seguinte equação continua verdadeira:

$\Diamond x = \sim\Box\sim x$

Assim, a proposição "é possível que fantasmas sejam reais" é equivalente a "não é necessário que fantasmas não sejam reais".

Um aspecto importante da Lógica Modal é a diferença entre *verdade necessária* e *verdade contingente.* Para entender essa distinção, observe o exemplo a seguir:

Considere *S* = Ontem choveu em São Paulo.

Suponha que realmente tenha chovido em São Paulo ontem. Neste caso, ambas as proposições são verdadeiras:

S VERDADEIRA

$S \vee \sim S$ VERDADEIRA

Mesmo que as duas sejam verdadeiras, existe uma diferença entre os tipos de verdade. A primeira proposição é *contingencialmente verdadeira*, pois sua verdade é contingente ao que realmente aconteceu. Isto é, a situação poderia ter sido diferente. É possível até que o boletim meteorológico tenha se equivocado.

A segunda proposição, porém, é *necessariamente verdadeira*. Isto é, independentemente das circunstâncias em São Paulo ontem, há uma necessidade lógica de que essa proposição seja verdadeira.

A graduação dessa diferença se torna ainda mais clara depois que adiciono o operador de necessidade:

$\Box S$ FALSO

$\Box S \vee \sim S$ FALSO

Nesse caso, a primeira proposição diz: "é necessário que tenha chovido ontem em São Paulo". Na Lógica Modal, essa proposição é falsa, pois pode haver uma situação em que não tenha chovido lá ontem. Por outro lado, a segunda proposição diz: "é necessário que tenha chovido ou não em São Paulo ontem". Essa proposição é verdadeira, ressaltando o nível mais alto de verdade independente de acontecimentos do mundo real.

Lidando com proposições de discurso indireto

A Modal é apenas um dos tipos de Lógica que tenta incluir proposições de *discurso indireto* que não podem ser traduzidas para a LS ou para a LQ.A *Lógica Deôntica* permite que você trate de proposições de obrigação e permissão. Por exemplo:

É *obrigatório* parar no sinal vermelho.

É *permitido* seguir no sinal verde.

De forma semelhante, *a Lógica Epistêmica* inclui operadores para lidar com proposições que tratam de conhecimento e opinião. Por exemplo:

Arnie *sabe que* Beth está esperando por ele.

Arnie *acha que* Beth está esperando por ele.

Levando a Lógica para uma Ordem Superior

Lembre-se de que a Lógica Quantitativa (LQ) contém constantes individuais, variáveis e constantes de predicado, mas não possui variáveis de predicado. Isso lhe permite quantificar indivíduos, mas não os predicados. Por exemplo, você pode representar a seguinte frase:

Todos os banqueiros são ricos.

como

$\forall x\ [Bx \rightarrow Rx]$

Para ser específico, essa proposição lhe diz que tudo que tem a propriedade de ser banqueiro também tem a de ser rico.

Porém, depois que você passa a enfocar os predicados, pode discuti-los de forma lógica. Por exemplo, suponha que esteja contratando um assistente e procure por um que seja simpático, inteligente e otimista. Na LQ é muito simples determinar as constantes para esses predicados:

Considere F = x é simpático.

Considere I = x é inteligente.

Considere U = x é otimista.

Então, você pode representar a seguinte frase:

Nádia é inteligente e Jason é otimista.

como

$In \wedge Uj$

No entanto, na LQ você não pode representar esta proposição:

Nádia tem todos os predicados de Jason.

LEMBRE-SE

A *Lógica de Segunda Ordem* (também chamada de *Predicativa de Segunda ordem*) permite que você lide com proposições sobre os predicados dos indivíduos, quantificando as variáveis de predicado, ao invés de apenas as individuais. Por exemplo, suponha que *X* signifique determinado predicado. Usando essa variável, você pode representar a proposição anterior como:

$\forall X\,[Xj \rightarrow Xn]$

A tradução literal dessa proposição é “para todo o predicado *X*, se Jason o tem, Nádia também tem”.

Você ainda pode quantificar variáveis individuais. Por exemplo, considere a seguinte proposição:

Alguém tem todas as propriedades que Jason tem.

Pode representar essa proposição como:

$\forall X\,\exists y\,[Xj \rightarrow Xy]$

A tradução literal dessa representação é “para todo o predicado *X*, existe um indivíduo *y* tal que, se Jason tiver *X*, então *y* também terá *X*”.

Indo Além da Consistência

De certa maneira, *a Lógica Paraconsistente* é o avesso da Multivalorada. Enquanto na Multivalorada uma proposição pode ser *nem verdadeira nem falsa*, na paraconsistente ela pode ser *tanto verdadeira quanto falsa*. Isto é, toda proposição tem ao menos um, mas possivelmente dois valores: **V** e/ou **F**.

Em outras palavras, é possível que a proposição

$x \wedge \sim x$

seja verdadeira. Colocando de outra maneira, a Lógica Paraconsistente lhe permite infringir a lei da não contradição, admitindo que uma proposição seja tanto verdadeira quanto falsa. Veja o Capítulo 1 para saber mais sobre a lei da não contradição.

Essa diferença essencial entre a Lógica Paraconsistente e a Clássica também cria novos obstáculos em uma importante regra de inferência — o silogismo disjuntivo (**SD**). Para refrescar sua memória, veja o Capítulo 9. Observe essas constantes:

Considere *A* = Ken mora em Albuquerque.

Considere *S* = Ken mora em Santa Fé.

Na Lógica Clássica, o **SD** permite que você faça a seguinte inferência:

$A \vee S, \sim A : S$

Esse argumento diz: quer Ken more em Albuquerque, quer em Santa Fé, se ele não mora em Albuquerque, então mora em Santa Fé.

Porém, na Lógica Paraconsistente, essa inferência não é verdadeira. Embora essa falha da **SD** na Lógica Paraconsistente possa parecer um pouco estranha a princípio, ela faz todo sentido se você parar para pensar no porquê.

Suponha que tanto *A* quanto *~A* sejam verdadeiras, o que é permitido na Lógica Paraconsistente, sem importar o quanto estranho isso possa parecer. Isto é, Ken tanto *mora* quanto *não mora* em Albuquerque. Nesse caso, a proposição

$A \vee S$

é verdadeira mesmo que S seja falsa. Como acabei de falar, a proposição

$\sim A$

também é verdadeira. Nesse caso, ambas as proposições são verdadeiras, mas a proposição

S

é falsa. Em outras palavras, as duas premissas são verdadeiras e a conclusão é falsa, então, a **SD** é um argumento inválido na Lógica Paraconsistente.

Assim, a Lógica Paraconsistente permanece consistente mesmo que, por definição, seja inconsistente. Colocando de outra maneira, a Lógica Paraconsistente é tanto consistente quanto inconsistente — isso, se você parar para pensar, é o resumo de toda a ideia; ou não.

Dando um Salto Quântico

Você já viu um charlatão fazendo o jogo das conchas? Ele pega uma ervilha, coloca-a na mesa e esconde-a sob umas das conchas. Depois, ele coloca mais duas conchas vazias ao lado da primeira e, em movimentos habilidosos, move as três conchas sobre a mesa. Se você conseguir adivinhar em que concha a ervilha está escondida, ganha.

Esse truque engenhoso parece fácil, mas tem feito muita gente perder dinheiro por aí. Geralmente, o charlatão tem uma incrível habilidade com as mãos e consegue esconder a ervilha na mão e depois sob uma outra concha, sem ser notado.

As partículas subatômicas também parecem funcionar de acordo com um jogo de conchas cósmico que é fácil de descrever, mas muito difícil de compreender. Essas partículas são a matéria-prima de tudo o que existe no universo — incluindo você, eu e este livro. Quanto mais os cientistas descobrem sobre o funcionamento do Universo nesse nível submicroscópico, mais estranho ele fica.

Apresentando a Lógica Quântica

Um dos aspectos mais estranhos no modo como o Universo se comporta nesses minúsculos níveis é que neles a Lógica entra em colapso. *Por que* isso acontece é o que todos tentam adivinhar, embora alguns físicos teóricos estejam chegando bem perto dessa resposta. Mas *o que* acontece já está muito bem documentado e é descrito pela *Lógica Quântica*.

Enquanto eu o apresento à Lógica Quântica, apenas tenha em mente duas coisas:

> ✓ A Lógica Quântica é ciência, não ficção científica — inúmeros experimentos científicos já a demonstraram como sendo a forma pela qual as coisas funcionam.
>
> ✓ Ela simplesmente não faz sentido.

À medida que a descrevo, não pense que não está conseguindo entender uma coisa que o resto do mundo entende. Ninguém compreende *por que* o universo funciona dessa maneira. Sendo assim, apenas concentre-se no *que* acontece e será o suficiente.

O jogo das conchas

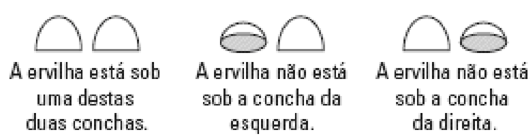
Imagine um jogo das conchas com apenas duas conchas, jogado com partículas subatômicas, que chamaremos de *ervilhas.* Nesse universo, é a Lógica Quântica que governa. Agora comece entendendo que a seguinte proposição é verdadeira:

> **Proposição 1:** A ervilha está sobre a mesa e sob qualquer das duas conchas, a da direita ou a da esquerda.

Na Lógica Quântica, muitas das suposições básicas da Lógica Clássica permanecem intactas. Uma proposição pode ter um dentre dois valores possíveis: **V** ou **F**. Os operadores básicos da LS funcionam da mesma maneira aqui. Dessa forma, na Lógica Quântica, assim como na LS, você pode declarar variáveis como essas, conforme ilustrada na Figura 21-1:

> Considere *P* = A ervilha está sobre a mesa.
>
> Considere *Q* = A ervilha está sob a concha da esquerda.
>
> Considere *R* = A ervilha está sob a concha da direita.

Figura 21-1: Um jogo desconcertante.



Com essa estrutura colocada, você pode escrever uma proposição em português e traduzi-la em símbolos. Por exemplo, a Proposição 1, traduzida para a Lógica Quântica, fica assim:

$$P \wedge (Q \vee R)$$

Até aqui tudo bem: tudo até agora é igual à LS. Agora, se você estivesse usando a LS, usaria a lei distributiva (veja o Capítulo 10) para escrever uma proposição equivalente. Mas a lei distributiva não se aplica à Lógica Quântica, de modo que você não pode reescrever a proposição anterior da seguinte forma:

$(P \wedge Q) \vee (P \wedge R)$ ERRADO!

Mas antes de deixar essa proposição de lado, observe que ela é traduzida de volta para o português da seguinte forma:

> **Proposição 2:** Ou a ervilha está sobre a mesa e sob a concha esquerda, ou a ervilha está sobre a mesa e sob a concha direita.

Em outras palavras, a Proposição 1 e a Proposição 2 não são necessariamente equivalentes, nem uma implica necessariamente na outra. Então, é possível que a Proposição 1 seja verdadeira e que a Proposição 2 seja falsa. Ou seja, é verdadeiro que a ervilha esteja sob uma das duas conchas, mas não está sob a concha esquerda e não está sob a concha direita.

Como você pode ver nesse exemplo, a Lógica Quântica contradiz, fundamentalmente, a LS. Ela também contradiz tudo o que parece possível, normal e sensato. Mas essa é a situação. As partículas que compõem o universo obedecem a essas leis. Assustador, não é mesmo?

Capítulo 22

Paradoxo e Sistemas Axiomáticos

Neste Capítulo

- Entendendo a Teoria dos Conjuntos e o Paradoxo de Russell
- Como o sistema axiomático da LS atende às expectativas
- Avaliando consistência e completude
- Limitando a matemática com o Teorema de Incompletude de Gödel

Considere a seguinte proposição:

Esta proposição é falsa.

Se a proposição é verdadeira, então tem que ser falsa. No entanto, se for falsa, tem que ser verdadeira. Esse problema, chamado de *Paradoxo do Mentiroso*, surgiu na Grécia antiga.

À primeira vista, não parecem nada mais que curiosidade mas, sob muitas formas, paradoxos como esse têm surgido e ressurgido, causando inúmeros problemas para os estudiosos da Lógica e desafiando-os a procurar maneiras de resolvê-los.

Neste capítulo, explico o Paradoxo de Russell (uma modificação do Paradoxo do Mentiroso), que obrigou os lógicos a fazerem uma reestruturação radical nos alicerces da Teoria dos Conjuntos e da Lógica. O que leva a uma discussão sobre o *Principia Mathematica*, que é a tentativa de elaborar a Teoria dos Conjuntos, a Lógica e, posteriormente, toda a Matemática baseada em um conjunto de suposições chamadas de axiomas. Você ainda verá como a Lógica se sai nos derradeiros testes da certeza matemática e, quando eu o apresentar ao teorema da incompletude de Gödel, dará, finalmente, uma olhada nos limites do que pode ser provado logicamente.

Fundamentando a Lógica na Teoria dos Conjuntos

A formulação da Lógica de Gottlob Frege, no final do século XIX, era fundamentada no relativamente novo trabalho de Georg Cantor, chamado de *Teoria dos Conjuntos*, que trazia uma maneira incrivelmente simples de organizar objetos no mundo real, mas também fornecia um modo unificado de definir objetos matemáticos, tais como números, através de suas propriedades matemáticas.

Neste tópico, você verá como a Teoria dos Conjuntos fornece o alicerce natural da Lógica, como ele foi ameaçado e se solidificou novamente.

Estabelecendo as coisas

Essa teoria trata obviamente de *conjuntos*, que são simplesmente uma coleção de coisas. Por exemplo:

> Considere *S* = o conjunto de todas as camisas que eu tenho.
>
> Considere *H* = o conjunto de todos os chapéus que você tem.

Um conjunto só é corretamente definido quando você pode distinguir claramente o que está *nele* e o que *não* está.

Os itens de um determinado conjunto são chamados de *elementos*. Por exemplo, a camisa que estou usando agora é um elemento do conjunto *S*, e o seu chapéu favorito (supondo que você tenha um) é um elemento do conjunto *H*.

Conjuntos podem conter outros conjuntos, que são chamados de *subconjuntos*. Por exemplo:

Considere B = o conjunto de todas as camisas azuis que tenho.

Considere L = o conjunto de todas as camisas que tenho e que estão no cesto de roupa suja.

Tanto B quanto L são subconjuntos de S. Isto é, qualquer elemento que estiver em qualquer um desses conjuntos estará também no S.

LEMBRE-SE

Ainda que a Teoria dos Conjuntos possa parecer um tanto simplista, ela é uma forma poderosa de expressar ideias lógicas.

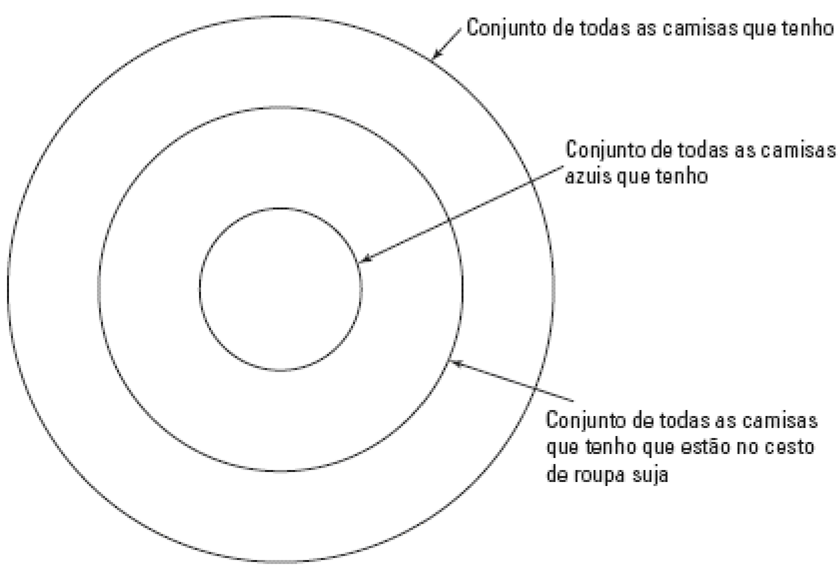
Por exemplo, considere a seguinte proposição

Todas as minhas camisas azuis estão no cesto de roupa suja.

Essa proposição é facilmente expressada pela Lógica Quantitativa (LQ):

$\forall x \, [Bx \rightarrow Lx]$

Essa proposição é verdadeira, se e somente se o conjunto B for um subconjunto do L :



Embora a Teoria dos Conjuntos e a Lógica pareçam diferentes na superfície, ambas expressam ideias semelhantes. É por isso que Frege fundamentou sua Lógica no solo aparentemente sólido da Teoria dos Conjuntos. Enquanto ela se mantivesse livre de contradições, a Lógica também seria considerada consistente. Essa teoria era tão simples que parecia inatacável.

Transtorno no paradoxo: Reconhecendo o problema com a Teoria dos Conjuntos

Foi preciso um gênio, Bertrand Russell, para visualizar o problema com a Teoria dos Conjuntos. Em sua homenagem, essa falha foi denominada *Paradoxo de Russell*, que se fundamenta na noção de *autorreferência.* O paradoxo do mentiroso é um exemplo de paradoxo de autorreferência: a raiz do problema é que a frase fala sobre si mesma.

A Teoria dos Conjuntos enfrentou um problema semelhante, pois era possível que um conjunto contivesse a si mesmo como elemento. Na maioria das vezes, porém, conjuntos não contêm a si mesmos como elementos. Ao examinar novamente o exemplo das camisas e dos chapéus, mencionado no tópico anterior, você verá que o conjunto *S* contém somente camisas e o *H* apenas chapéus. Mas observe este conjunto:

> Considere *X* = o conjunto de todos os conjuntos mencionados neste capítulo.

É claro que o conjunto *X* contém os conjuntos *S, H, B* e *L* como elementos, mas também contém *a si mesmo* como um deles, pois o conjunto *X* é mencionado neste capítulo.

Isso não é um problema por si só, mas gera um rapidamente. Considere o que aconteceria se você definisse este conjunto:

> Considere *Z* = o conjunto de todos os conjuntos que não contêm a si mesmo como elementos.

Nesse caso, os conjuntos *S, H, B* e *L* são elementos do conjunto *Z*, mas o conjunto *X* não é. Aqui está o problema:

> O conjunto *Z* é um elemento de si mesmo?

O problema aqui é semelhante ao paradoxo do mentiroso: se o conjunto *Z* é um elemento de si mesmo, então, por definição, ele *não é* um elemento de si mesmo. Consequentemente, se o conjunto *Z* não é um elemento de si mesmo, então, por definição, ele *é*.

Desenvolvendo uma solução no Principia Mathematica

O Paradoxo de Russell (veja o tópico anterior) foi mais do que simplesmente um problema (Frege teria ficado apavorado com a ideia de que todo o seu trabalho havia sido destruído). Na verdade, o paradoxo obrigou os lógicos a reformularem a Teoria dos Conjuntos e a Lógica de um modo diferente. A dificuldade estava em descobrir como evitar a intromissão do paradoxo e, ao mesmo tempo, manter a magnitude de tudo o que era útil e descritivo nos sistemas originais.

Em uma tentativa de resolver o paradoxo e criar uma base sólida para a Matemática, na primeira década do século XX, Bertrand Russell e Alfred North Whitehead escreveram o *Principia Mathematica.* Esse ambicioso trabalho foi a primeira tentativa consistente de descrever toda a Matemática como uma *sistema axiomático* formal — uma organização das noções da Matemática baseada em um pequeno número de proposições presumidas como verdadeiras.

LEMBRE-SE

A essência de um sistema axiomático é uma breve lista de proposições simples chamadas de *axiomas*, que são combinados de modos específicos para originarem um conjunto muito maior de proposições chamados de *teoremas*. Russell e Whitehead escolheram cuidadosamente seus axiomas, tendo em vista várias finalidades:

- ✓ Criar um sistema poderoso o bastante para originar proposições sofisticadas na Matemática, como os teoremas.
- ✓ Evitar *todas* as inconsistências, tais como o Paradoxo de Russell.
- ✓ Mostrar que *todas* as verdades matemáticas possíveis poderiam decorrer deles, sob a forma de teoremas.

Russell e Whitehead conseguiram atingir seu primeiro objetivo. O sistema criado por eles eliminou os paradoxos de autorreferência, como o Paradoxo de Russell. No entanto, não puderam concluir se o Principia Mathematica conseguiria afastar *todas* as inconsistências e criar um método que originasse *toda* a matemática.

Muito embora o conjunto de axiomas do *Principia* tenha solucionado o problema do Paradoxo de Russell, na prática ele era desajeitado e não foi difundido entre os matemáticos. Ao invés disso, um diferente conjunto de axiomas, os *Zermelo-Frankel* (axiomas ZF), resolveu a questão, distinguindo os conjuntos de objetos com definições mais imprecisas, conhecidos como *classes*.

Hoje, a expressão *Teoria dos Conjuntos*, se refere, em geral, a uma das várias versões dessa teoria baseada nos axiomas ZF. Por outro lado, existem versões mais simples dessa teoria que não tentam evitar os paradoxos. Elas são todas juntas chamadas de *Teoria Ingênua dos Conjuntos.*

Entender como os sistemas axiomáticos funcionam é o principal objetivo deste capítulo. Discutirei esse conceito em maiores detalhes nos tópicos seguintes.

Descobrindo o Sistema Axiomático para a LS

Para que você entenda como os teoremas são formalmente derivados na LS, este tópico mostra a estrutura básica dos sistemas axiomáticos que Russell e Whitehead desenvolveram para o *Principia Mathematica*.

Resumindo, um sistema axiomático formal tem quatro requisitos, que são dispostos em conjuntos, pois na Lógica e em toda a Matemática, no sentido formal, *tudo* está estabelecido em termos de conjuntos. É por isso que foi tão importante aparar as arestas da Teoria dos Conjuntos, como expliquei no tópico "Fundamentando a Lógica na Teoria dos Conjuntos".

Esses quatro requisitos são

- ✓ **Requisito 1:** Um conjunto de símbolos
- ✓ **Requisito 2:** Um conjunto de regras para decidir que série de símbolos são formulas bem formadas (FBF)
- ✓ **Requisito 3:** Um conjunto de axiomas
- ✓ **Requisito 4:** Um conjunto de regras para combinar axiomas e/ou teoremas para criar novos teoremas

Assim, com esses requisitos em mente, você pode ver como a LS se encaixa bem na definição de um sistema axiomático. O Requisito 1 é atendido, pois a LS contém um conjunto de símbolos — operadores, constantes e parênteses (veja o Capítulo 4). O Requisito 2 é preenchido pelo conjunto de regras para as FBFs (veja o Capítulo 14).

Para satisfazer ao Requisito 3, aqui estão os quatro axiomas da LS do *Principia Mathematica*:

1. $(x \vee x) \rightarrow x$

2. $x \rightarrow (x \vee y)$
3. $(x \vee y) \rightarrow (y \vee x)$
4. $(x \vee y) \rightarrow ((z \vee x) \rightarrow (z \vee y))$

Quanto ao Requisito 4, observe as duas regras seguintes para a elaboração de novos teoremas na LS:

> ✓ **Regra da Substituição:** Em todos os casos, você pode substituir uma constante ou uma proposição inteira por uma variável, desde que seja de forma uniforme.
>
> Por exemplo, duas substituições possíveis para o axioma 1 são
>
> $(P \vee P) \rightarrow P$
>
> $((P \wedge Q) \vee (P \wedge Q)) \rightarrow (P \wedge Q)$
>
> ✓ **Modus Ponens (MP):** Se você sabe que $x \rightarrow y$ e x são teoremas, pode adicionar y à sua lista de teoremas. Veja o Capítulo 9 para saber mais sobre a Modus Ponens.

A partir dessa breve lista de regras é possível criar todas as regras de inferência para a LS, que foram discutidas na Parte III. Isso lhe mostra que mesmo que a lista de axiomas seja pequena, os teoremas originados deles são muito poderosos.

Provando Consistência e Completude

Com a formalização da LS como um sistema axiomático, surgiram duas provas importantes relacionadas a esta: todo teorema é uma tautologia e toda tautologia é um teorema. Isto é, teoremas e tautologias são equivalentes na LS.

Ao longo deste livro você usa tanto as tabelas verdade quanto as provas indiscriminadamente para elaborar conclusões sobre as proposições da LS. Depois que sabe que todo teorema é uma tautologia e vice-versa, você pode aplicar os dois métodos em um determinado problema: o sintático (provas) e o semântico (tabelas verdade). Com eles pode testar se um teorema específico é uma tautologia, construindo uma tabela verdade e verificá-lo com o método apresentado no Capítulo 6.

Ainda assim, se você pensar um pouco perceberá que a equivalência entre teoremas e tautologias na LS não deve ser ignorada. Por que não acontece, por exemplo, de um teorema não ser uma tautologia? Ou, o contrário, por que não existe uma tautologia que não possa ser elaborada como um teorema, usando os axiomas finitos e as regras do tópico anterior?

A primeira questão trata da consistência da SL e, a segunda de sua completude. Discuto ambas nos tópicos seguintes.

Consistência e completude da LS e da LQ

Em 1921, o matemático Emil Post provou que a LS é consistente e completa. Um sistema axiomático será *consistente* se e somente se todo teorema gerado por ele for uma tautologia; e será *completo* se e somente se toda tautologia puder ser elaborada como um teorema.

Embora inconsistência pareça perigosa para os sistemas axiomáticos, ameaçando minar até o trabalho mais bem construído (por exemplo, a Lógica de Gottlob Frege), a completude parecia mais com o Santo Graal. Em outras palavras, embora a inconsistência fosse algo com que os matemáticos e lógicos tinham que se preocupar, a completude era impossível de ser alcançada.

Na verdade, a completude era o objetivo secreto dos matemáticos desde a época dos gregos. Euclides, por exemplo, é considerado o fundador da Geometria, mesmo ela tendo sido estudada milhares de anos antes dele (veja o Capítulo 2 para saber mais sobre Euclides). A grande descoberta de Euclides foi que a Geometria poderia ser baseada em cinco axiomas e que, a partir deles, todas as outras proposições verdadeiras sobre a Geometria poderiam ser elaboradas como teoremas.

A década que sucedeu a prova de Emil Post foi uma época frutífera para a Lógica. Em 1928, David Hilbert e William Ackerman provaram que a Lógica Quantitativa (LQ) é consistente. Então, em 1931, Kurt Gödel provou que a LQ é completa. Além disso, esse importante resultado foi sua tese de doutorado e o primeiro trabalho daquele que se tornaria um dos maiores matemáticos do século XX.

Formalizando a Lógica e a Matemática com o Programa de Hilbert

Nos anos 1920, a Lógica e a Matemática se desenvolveram o suficiente para gerar uma análise precisa sobre a possibilidade de toda a verdade matemática ser demonstrada como um teorema. O matemático David Hilbert colaborou na defesa da formalização completa da Lógica e da Matemática. Essa formalização passou a ser conhecida como *Programa de Hilbert*.

Hilbert percebeu que aquilo que os filósofos e matemáticos haviam intuitivamente buscado desde a época de Aristóteles e Euclides estava, agora, potencialmente ao alcance: um sistema axiomático único, livre de inconsistências, para expressar e calcular todas as verdades lógicas e matemáticas.

O Programa de Hilbert reforçou a necessidade de colocar toda a matemática em termos axiomáticos rígidos. Todas as suposições lógicas devem ser expressas de maneira explícita, em linguagem formal, sem qualquer ambiguidade (o *Principia Mathematica* foi um exemplo dessa tentativa de formalização e Hilbert estudou-o em detalhes). A partir desse alicerce, a noção intuitiva de prova matemática pôde ser formalizada e resultou na *Teoria da Prova.*

Axiomas de Peano

Existe uma importante diferença entre os axiomas do *Principia Mathematica* e os da Lógica: os do *Principia* eram sólidos o bastante para originar proposições matematicamente sofisticadas (veja o tópico "Desenvolvendo uma solução no *Principia Mathematica*" para saber mais sobre ele).

Os axiomas do *Principia* tornaram possível a elaboração dos cinco axiomas fundamentais da Teoria dos Números, a base de toda a matemática superior —, desenvolvida pelo matemático Giuseppe Peano:

1. Zero é um número.
2. Se *a* é um número, então, o sucessor de *a* é um número.
3. Zero não é o sucessor de um número.
4. Dois números cujos sucessores são iguais são iguais entre si.
5. Se um conjunto *S* contém zero e o sucessor de cada número, então, cada número está contido em *S* (isso é chamado de *Axioma de Indução*).

Em 1931, Gödel demonstrou que o *Principia Mathematica*, um sistema axiomático poderoso o bastante para originar os axiomas de Peano (e, assim, padronizar a matemática), estava condenado a ser inconsistente ou incompleto. De modo geral, ele demonstrou que *todo* sistema axiomático que fosse poderoso o bastante para padronizar a matemática estaria condenado à mesma sorte.

Hilbert venceu a eterna busca pelas provas de consistência e completude dos sistemas axiomáticos. Comparando um sistema axiomático a um navio, você pode dizer que a consistência significa que o navio não vai afundar e que a completude significa que o navio o levará para onde você quer ir.

Teorema da Incompletude de Gödel

A prova de que a LQ era consistente e completa deixou os matemáticos otimistas quanto ao sucesso do Programa de Hilbert. Ironicamente, o homem que provou a completude da LQ demonstrou, pouco tempo depois, que o Programa de Hilbert nunca teria êxito.

Em 1931, Kurt Gödel publicou seu Teorema da Incompletude, que afirma que nenhum sistema axiomático pode ter todas as três propriedades listadas a seguir:

> ✓ **Consistência:** Todo teorema do sistema é uma tautologia na área que o sistema pretende padronizar.
>
> ✓ **Completude:** Toda tautologia na área em que o sistema pretende padronizar é um teorema do sistema.
>
> ✓ **Força suficiente para padronizar a matemática:** O sistema pode ser aplicado como um modelo para a matemática.

A importância do teorema de Gödel

Esse teorema é geralmente considerado o mais importante resultado matemático do século XX. Ele é impressionante porque põe limites naquilo que a matemática pode fazer. Mais exatamente, põe limites nos graus em que os sistemas axiomáticos podem descrever a matemática. Como um golpe de misericórdia, o Teorema de Incompletude de Gödel demonstra que o objetivo do Programa de Hilbert (veja o tópico anterior) era inatingível.

Gödel provou sua conjectura adotando uma estratégia na mesma linha dos Paradoxos do Mentiroso e de Russell: a autorreferência. Ao invés de se frustrar com os paradoxos, Gödel fez com que eles trabalhassem a seu favor. Sua estratégia se baseava no fato de que todo sistema que fosse expressivo o bastante para padronizar a matemática complexa também poderia padronizar a si mesmo — uma tarefa extremamente difícil.

Como ele fez isso

Como primeiro passo, Gödel demonstrou como atribuir um único número, chamado de *número de Gödel*, para todas as séries de símbolos de um sistema. Essa numeração permitiu que ele numerasse de maneira única não apenas séries aleatórias, mas também proposições, teoremas e até argumentos inteiros, válidos ou não. Por exemplo, todas essas séries poderiam ser representadas assim:

> 4 + 9 = 13
>
> 4 + 9 = 1962
>
> $\forall x \exists y [x + y = 3]$
>
> = 48 < + 33- = 7 =

Já que a numeração de Gödel funcionou no nível das cadeias, literalmente nada que pudesse ser expresso em sistemas lhe escapou, incluindo a última cadeia, que não faz sentido.

Ele depois demonstrou como usar o sistema para construir proposições especiais, chamadas de *metaproposições*, que se referem a outras proposições. Por exemplo:

> A proposição “4 + 9 = 13” é um teorema.
>
> A proposição “4 + 9 = 13” não é um teorema.
>
> A proposição “4 + 9 = 13” inclui o símbolo “3”.
>
> Existe uma prova de que a proposição “4 + 9 = 13” é um teorema.

Essas metaproposições sozinhas eram apenas proposições, cada uma com seu próprio número de Gödel. Dessa forma, uma metaproposição poderia fazer referência a outra e até a si própria. Por exemplo:

“Não existe prova de que essa proposição seja um teorema.”

LEMBRE-SE

O paradoxo contido na proposição anterior é ainda mais sutil do que parece, pois Gödel construiu a proposição para garantir que no nível da semântica ela seja uma tautologia (a matemática por trás dessa garantia é completamente inatingível, então apenas aceite minha palavra de que isso é verdade). Assim, considere o seguinte:

- ✓ Se a proposição *pode* ser elaborada como um teorema, então ela não é um teorema, de modo que a contradição da proposição também pode ser elaborada em um teorema, o que torna o sistema *inconsistente* por definição.
- ✓ Se a proposição *não pode* ser elaborada como um teorema, então, o sistema é, por definição, *incompleto.*

Se isso parece complicado é porque realmente é. Eu apenas tratei superficialmente do problema. Entender *o que* Gödel realizou é fascinante, mas compreender *como* ele fez isso é algo que mesmo a maioria dos matemáticos só consegue alcançar de um modo mais ou menos superficial.

Refletindo sobre o Significado de Tudo Isto

Desde que Gödel publicou sua prova, que acabou com a utilidade dos sistemas axiomáticos para expressar as verdades matemáticas, as opiniões se dividiram sobre seu significado em um nível filosófico.

De certo modo, o Teorema da Incompletude de Gödel foi a resposta ao sonho de 250 anos de Leibniz de encontrar um sistema de Lógica poderoso o suficiente para calcular questões envolvendo direito, política e ética (veja o Capítulo 2 para saber mais sobre Leibniz). A resposta de Gödel foi um sonoro e definitivo: "Não! Não se pode fazer isso!" Supondo que a Lógica seja insuficiente para abranger um modelo completo de matemática, certamente é improvável que algum dia se consiga fornecer as ferramentas para resolver questões éticas por mero cálculo.

Você pode pensar que a prova de Gödel implica que a mente racional é limitada em sua habilidade de entender o universo. Mas embora a mente possa ter suas limitações, o resultado de Gödel não comprova que elas existam. A prova apenas afirma como os sistemas axiomáticos são limitados para serem usados para padronizar outros tipos de fenômenos. A mente, porém, pode possuir um número muito superior de capacidades do que um sistema axiomático ou uma Máquina de Turing.

Outra reação comum e provavelmente precipitada ao trabalho de Gödel é presumir que sua prova implica em um limite para a inteligência artificial. Afinal de contas, a inteligência humana tem se desenvolvido muito bem nesse universo. Por que outras formas de inteligência, até mesmo artificiais, não podem ser desenvolvidas nos mesmo parâmetros?

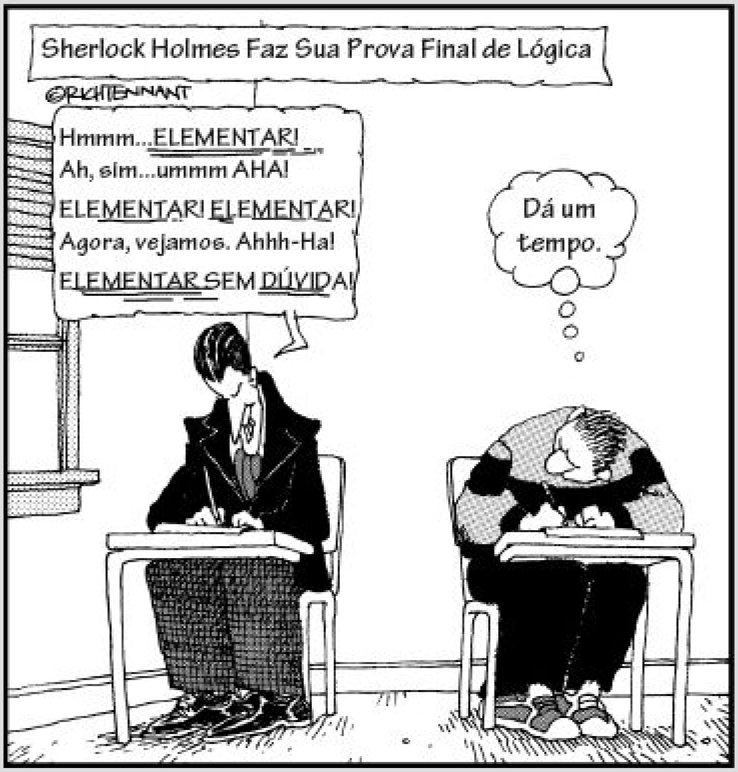
Como qualquer resultado científico fantástico do século XX, tal como a Teoria da Relatividade e a Mecânica Quântica, a prova de Gödel responde a um

LEMBRE-SE

conjunto de questões somente para abrir novas questões ainda mais atraentes.

Parte VI
A Parte dos Dez

A 5ª Onda Por Rich Tennant



Nesta parte...

Quem não gosta de listas dos dez mais? Bem, espero que você goste, pois esta parte traz três listas dos dez mais que lhe dão informações divertidas sobre a Lógica — algumas poderão ser bem úteis na hora de fazer seu próximo exame de Lógica!

O Capítulo 23 mostra minhas dez citações favoritas de pensadores e cínicos sobre a Lógica através dos tempos. No Capítulo 24, apresento meus indicados para os dez melhores lógicos. O Capítulo 25 contém dez super dicas para ter sucesso em um exame de Lógica.

Capítulo 23
Dez Citações sobre Lógica

Neste Capítulo
- ► Citações Lógicas
- ► Assumindo uma perspectiva lógica

Está bem, na verdade, há 11 citações — mas quem está contando? Elas são todas interessantes e oferecem uma variedade de perspectivas sobre a Lógica.

"Lógica é apenas o princípio da sabedoria e não seu fim."

Leonard Nimoy — Ator americano (falando como Spock de *Jornada nas Estrelas*)

"Lógica: a arte de pensar e raciocinar em estreita concordância com as limitações e incapacidades da incompreensão humana."

Ambrose Bierce — Escritor/sátiro americano

"Lógica é a anatomia do pensamento."

John Locke — Filósofo inglês do século XVII

"Lógica pura é a ruína do espírito."

Antoine de Saint-Exupery — Escritor francês

"Lógica é a arte de errar com confiança."

Joseph Wood Krutch — Naturalista/escritor americano

"Lógica é como a espada — quem faz uso dela perecerá por ela."

Samuel Butler — Romancista/ensaísta inglês

"Você só pode encontrar a verdade com a Lógica se já a tiver encontrado sem ela."

G. K. Chesterton – escritor inglês.

"Lógica leva você do ponto A para o B. A imaginação leva você para qualquer lugar."

Albert Einstein — Físico alemão

"Lógica cuida de si mesma; tudo que temos que fazer é observar como ela faz."

Ludwig Wittgenstein — Filósofo austríaco

"Lógica está nos olhos do lógico."

Gloria Steinem — Ativista/escritora americana

"Você pode usar a Lógica para justificar tudo. Esse é o seu poder e sua fraqueza."

Kate Mulgrew — Atriz americana (falando como sua personagem, a Capitã Kathryn Janeway em *Jornada nas Estrelas Voyager*)

Capítulo 24
Dez Grandes Nomes na Lógica

Neste Capítulo

- Como Aristóteles mudou tudo
- Descobrindo as pessoas que transformaram a Lógica
- Entendendo a evolução da Lógica

Eis os meus indicados para o Hall da Fama da Lógica. Muitas mentes brilhantes tiveram que ser deixadas de lado, ou essa lista teria quilômetros de comprimento, mas aqui estão os dez mais.

Aristóteles (384-322 a.C.)

Aristóteles foi o criador da Lógica. Antes dele, os filósofos (como Sócrates e Platão) e os matemáticos (como Pitágoras e Tales) apresentaram argumentos sobre uma ampla variedade de tópicos. Mas Aristóteles foi o primeiro a examinar a estrutura do próprio argumento.

LEMBRE-SE

Numa série de seis trabalhos filosóficos sobre Lógica, depois reunidos em um só, chamado de *Órganon*, Aristóteles identificou os conceitos fundamentais da Lógica. Ele definiu a proposição como sendo uma frase que possui veracidade ou falsidade (para saber mais sobre proposições, veja os Capítulos 1 e 3). Ainda estudou a estrutura dos argumentos válidos, chamados de silogismos (veja o Capítulo 2), que contêm premissas que levam inevitavelmente a uma conclusão. Séculos depois de sua morte, os trabalhos de Aristóteles sobre Lógica foram diversas vezes estudados e comentados, mas raramente superados (veja o Capítulo 2 para saber mais fatos curiosos sobre Aristóteles).

Gottfried Leibniz (1646-1716)

Um bem intencionado da Renascença, Gottfried Leibniz foi o primeiro filósofo da Era da Razão a ver o potencial da Lógica como ferramenta para o cálculo. Ele esperava que o cálculo lógico pudesse, algum dia, ser equiparado à matemática. Chegou a trabalhar com os princípios da representação simbólica da Lógica, antecipando a Lógica Formal em 200 anos (veja o Capítulo 2 para saber mais sobre Leibniz).

George Boole (1815-1864)

George Boole inventou a Álgebra Booleana, que foi o protótipo da Lógica Formal. A Álgebra Booleana foi o primeiro sistema de Lógica a usar o cálculo puro para determinar o valor verdade das proposições. Boole usou 1 para representar o verdadeiro e 0 para o falso. Cientistas da computação ainda usam variáveis booleanas como objetos que podem assumir esses dois valores e nenhum outro (veja o Capítulo 2 para saber mais sobre Boole e o Capítulo 14 para conhecer a Álgebra Booleana).

Lewis Carroll (1832-1898)

Mesmo sendo mais conhecido como o autor de *Alice no País das Maravilhas*, Lewis Carroll (cujo nome verdadeiro era Charles Dodgson) era professor de matemática da Cambridge University, na Inglaterra, e também escreveu diversos livros sobre Lógica. Ele ainda se divertia escrevendo quebra-cabeças de Lógica. Este é um de seus favoritos:

> Bebês não são lógicos.
>
> Ninguém despreza as pessoas que adestram crocodilos.
>
> Pessoas ilógicas são desprezadas.

O objetivo aqui é usar todas as três premissas para chegar a uma conclusão lógica, que nesse caso é "Bebês não podem adestrar crocodilos".

Para ser justo, Carroll não devia estar nesta lista uma vez que suas contribuições para a Lógica foram mais recreativas. Mas, de novo, ele era um lógico e, certamente, foi um grande nome, o que lhe permite chegar à conclusão de que ele era um grande nome da Lógica, logicamente falando.

Georg Cantor (1845-1918)

Georg Cantor foi o inventor da Teoria dos Conjuntos, que foi a base da Lógica e, pode se acreditar de toda a matemática (veja o Capítulo 2 para saber mais sobre Cantor e o Capítulo 22 para conhecer mais sobre a Teoria dos Conjuntos). Ele ainda foi o primeiro a incorporar à matemática a compreensão da infinitude como uma entidade calculável, ao invés de apenas um fantasma misterioso. Por todas essas contribuições e outras, Cantor é considerado por todos um dos maiores matemáticos do século XIX.

Gottlob Frege (1848-1925)

Gottlob Frege, inventor da Lógica Formal, construiu seu trabalho sobre o de Boole, Cantor e outros para desenvolver os primeiros sistemas lógicos que, posteriormente se tornariam a Lógica Sentencial e a Quantitativa. A Lógica de Frege incluiu os cinco operadores lógicos *não, e, ou, se* e *somente se*. Incluiu, ainda, símbolos para *qualquer que seja* e *existe.* Veja o Capítulo 2 para saber mais sobre Frege e suas contribuições para a Lógica.

Bertrand Russell (1872-1970)

Os quase cem anos de vida de Bertrand Russell incluíram notáveis realizações, como o Paradoxo de Russell e a coautoria de *Principia Mathematica*, com Alfred North Whitehead.

LEMBRE-SE

O Paradoxo de Russell resultou na reformulação da Lógica de Frege e da Teoria dos Conjuntos de Cantor — ambos são sistemas fundamentais que antes pareciam inatacáveis. O *Principia Mathematica* foi a tentativa de Russel para formular a matemática com perfeita consistência e completude lógica. Veja o Capítulo 2 para conhecer as discussões de Russell e seu lugar na história da Lógica e da Matemática.

David Hilbert (1862-1943)

David Hilbert foi tremendamente influente tanto na Lógica quanto na matemática. Ele defendia a redução rigorosa de toda a Matemática a axiomas (verdades autoevidentes) e teoremas (proposições que poderiam ser logicamente comprovadas através dos axiomas). Esse movimento da matemática ficou conhecido como Programa Hilbert, cujo objetivo era a criação de um sistema de axiomas para a Matemática que fosse tanto consistente quanto completo, isto é, que produzisse todos os teoremas verdadeiros possíveis e nenhum falso.

Embora Kurt Gödel tenha demonstrado que o objetivo do Programa Hilbert não era possível de ser alcançado, sua contribuição para o desenvolvimento da Lógica Matemática é inegável (veja o Capítulo 22 para saber mais sobre ele).

Kurt Gödel (1906-1978)

Gödel provou que a Lógica, em sua forma mais poderosa (Lógica Quantitativa), é matematicamente *consistente* e *completa*. Consistência significa que a Lógica não contém contradições. Completude significa que todas as proposições lógicas verdadeiras podem ser provadas por métodos sintáticos (veja o Capítulo 7 para a discussão sobre a prova sintática).

Porém, Gödel é mais conhecido por sua prova de que a matemática como um todo, *não* possui consistência e completude. Ele dizia que todo sistema matemático que não contivesse contradições deveria conter proposições que fossem verdadeiras e que não pudessem ser provadas usando os axiomas daquele sistema. Essa descoberta marcou o fim do Programa Hilbert e é considerada o mais importante resultado matemático do século

XX. Veja os Capítulos 2 e 22 para conhecer mais detalhes sobre o importante trabalho de Gödel.

Alan Turing (1912-1954)

Alan Turing provou que todos os cálculos realizados por humanos poderiam ser igualmente executados por computadores usando um conjunto específico de funções simples. Essas funções incluem a habilidade de verificar se certos tipos de condições são verdadeiras ou falsas e agir com base nesses resultados.

LEMBRE-SE

Turing chamava qualquer computador desse tipo de máquina universal de Turing (UTM na sigla em inglês). Tendo em vista que todo computador moderno é uma UTM e que a Lógica é o coração do processamento de dados de qualquer computador, ela é a pedra fundamental da Lógica Computacional (veja o Capítulo 20 para saber mais sobre como a Lógica e a Ciência da Computação trabalham juntas).

Capítulo 25

Dez Dicas para Ter Sucesso em um Exame de Lógica

Neste Capítulo

- Descobrindo as técnicas para ajudar na hora do exame de Lógica
- Encontrando o jeito mais fácil de fugir do "bloqueio"
- Entendendo a importância de verificar seu trabalho e admitir seus erros

Deixe-me adivinhar, você está lendo este livro freneticamente, oito horas antes de seu exame final de Lógica? Não se preocupe. Este breve capítulo traz dez dicas para que tenha sucesso no exame. Continue lendo, pois o que você aprenderá aqui vai gerar bons resultados!

Respire

Quando estiver sentado, esperando até que seu professor entregue o exame, inspire profundamente, contando lentamente até cinco e, então, expire da mesma forma. Repita o processo. Só isso. Faça assim por cerca de um minuto (não mais que isso — você não vai querer hiperventilar!). Você verá que ficará muito mais calmo.

Comece com uma Rápida Olhada em Todo o Exame

Vai levar apenas um minuto, mas olhar o exame inteiro coloca seu cérebro para trabalhar nos problemas o mais cedo possível, através de seu subconsciente. Assim, alguns problemas vão ser resolvidos de forma muito mais fácil durante o exame.

Essa técnica também lhe permitirá encontrar pistas úteis. Por exemplo, a questão 3 pode pedir uma definição de um termo que aparece nas questões 5, 6 e 7.

Faça um Aquecimento com um Problema Fácil Primeiro

Por que não se aquecer com um problema fácil? Atletas sabem a importância de se aquecer antes de colocar tudo para trabalhar. Garanto que o simples fato de escrever algo já fará com que você se sinta melhor.

Preencha as Tabelas Verdade, Coluna por Coluna

Você pode preencher uma tabela verdade do modo fácil ou do difícil. O fácil consiste em preenchê-la coluna por coluna, como aprendeu no Capítulo 6.

Se você tentar preenchê-la linha por linha, terá que trocar de “ferramenta” o tempo todo, o que fará com que demore mais para alcançar o mesmo resultado.

Se Ficar Travado, Anote Tudo o que Conseguir

As provas parecem tão claras no livro que alguns alunos pensam que têm que estar com tudo na cabeça, antes mesmo de começar.

Porém, fazer provas é um processo confuso, o que lhe permite ficar à vontade para agir como quiser. Use papel de rascunho para anotar todos os caminhos possíveis que passarem por sua cabeça. Depois que o grande “Aha!” acontecer, comece respondendo as questões cuidadosamente no papel que você pretender entregar.

Se Ficar REALMENTE Travado, Siga Adiante

Acontece: a resposta está na sua cara e você não consegue enxergar. Os minutos estão voando, seu coração está aos pulos e suas mãos estão tão suadas que o lápis está pingando.

Se você for do tipo que começa a rezar, esse é um bom momento para começar. Mas, lembre-se, o Senhor ajuda aqueles que se ajudam, então eu lhe direi: siga em frente!

É melhor perder uma questão do que cinco que você nem terá chance de fazer simplesmente porque ficou travado. Se tiver tempo, volte à questão. Quando voltar, a resposta vai pular na sua frente.

Se o Tempo For Curto, Termine as Coisas Mais Enfadonhas

De modo geral, tabelas verdade e árvores lógicas tendem a ser métodos de “preencher e interpretar”: se trabalhar lenta e progressivamente sempre chegará ao resultado. Por sua vez, as tabelas rápidas e as provas tendem a ser métodos criativos: eles podem ser mais rápidos se você estiver com a inspiração correta, mas não há garantias.

Então, quando o professor falar: “faltam dez minutos!”, deixe para lá aquela prova trabalhosa que simplesmente não está falando com você naquele momento e vá para as enfadonhas tabelas verdade que estava evitando até aquela hora. Quando o exame chega aos seus momentos finais é mais provável que você faça progressos nas tabelas em que estiver trabalhando do que se ficar apenas olhando para aquela prova impossível de entender.

Verifique Seu Trabalho

Sei que em alguns grupinhos verificar o exame antes de entregá-lo é considerado uma coisa antiquada e de gente esquisita — como carregar um lenço com um monograma no bolso ou tirar seu chapéu ao entrar em um prédio.

Mas pense na última vez que *não* checou seu exame antes de entregá-lo. Você realmente fez algo importante durante aqueles sete ou oito minutos que economizou? Não, provavelmente terá gasto esses preciosos minutos do lado de fora da sala com seus amigos, tagarelando sobre o exame.

Você simplesmente não odeia receber seu exame de volta, *depois* de avaliado, e olhar todos aqueles erros idiotas circulados em vermelho?

Tudo bem, estou chovendo no molhado aqui. Mas, mesmo que você encontre apenas um pequeno errinho de décimos de ponto, isso pode alavancar sua nota. Todavia, você pode conseguir notar 5, 10 ou até 20 errinhos desses, o que não é assim tão incomum.

Admita Seus Erros

Sei que esse conselho contraria todos os seus instintos de sobrevivência, mas escute o que estou lhe dizendo.

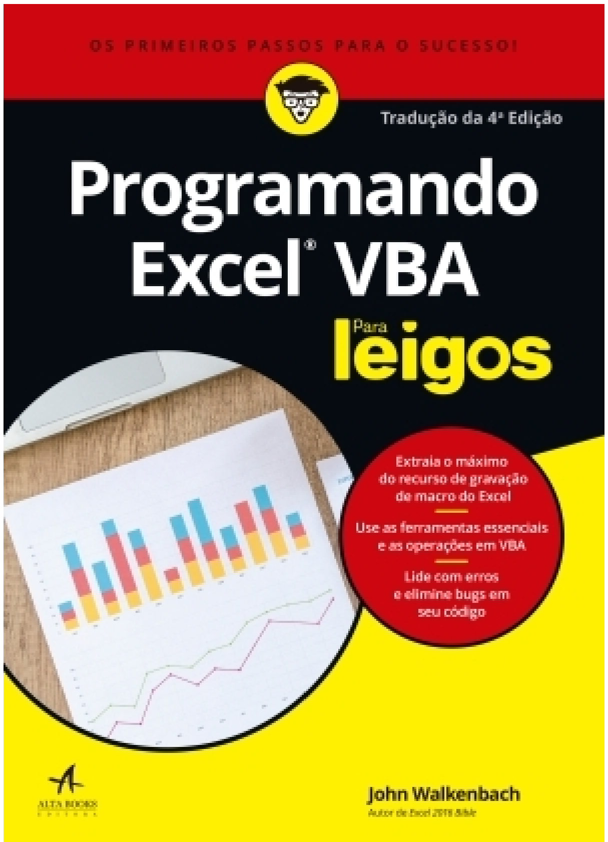
Por exemplo, considere esta história trágica: você suou por meia hora em cima de uma prova complicada tentando provar ($P \wedge Q$). Quando o professor começa a recolher o exame, chega na última linha e prova que ~($P \wedge Q$). Ai! Se você tivesse tempo, poderia procurar na prova, linha por linha, até encontrar o erro, mas tempo é exatamente o que você não tem.

Seu primeiro instinto é pensar: talvez o professor nem note e me dê a nota inteira — e está redondamente enganado (seu segundo instinto, que é abandonar a escola e mudar para um templo tibetano, também não parece uma boa ideia). Se você deixar o erro como está, o professor vai pensar que você não sabe a diferença entre uma proposição e sua negação.

Ao invés disso, circule o erro e escreva uma nota: "eu sei que está errado, mas não tive tempo de consertar". Assim, quando o professor ler sua resposta, terá algum contexto para avaliar. Se seu erro tiver sido pequeno, como esquecer do operador ~, provavelmente vai lhe dar alguns pontos pela questão. Mesmo que tenha errado feio, ao menos ele saberá que você percebeu o erro. Então, agora ele sabe que você não está totalmente perdido e que, também, é honesto. De qualquer modo você ganha!

Fique Até o Doloroso Final

Continue escrevendo até que o professor arranque o lápis dos seus dedos inchados. Você pode conseguir alguns pontinhos nesses últimos minutos e o professor perceberá seu esforço extra.


OS PRIMEIROS PASSOS PARA O SUCESSO!
Tradução da 4ª Edição
Programando Excel® VBA
Para leigos
Extraia o máximo do recurso de gravação de macro do Excel
Use as ferramentas essenciais e as operações em VBA
Lide com erros e elimine bugs em seu código
ALTA BOOKS EDITORA
John Walkenbach
Autor de Excel 2016 Bible


CELEBRANDO
20 ANOS
COMO O LIVRO
Nº 1 EM FINANÇAS
PESSOAIS
O que os Ricos Ensinam a Seus Filhos sobre Dinheiro
PAI RICO
PAI POBRE
NOVA EDIÇÃO ATUALIZADA E AMPLIADA — com 9 SEÇÕES DE ESTUDO INÉDITAS
ALTA BOOKS
ROBERT T. KIYOSAKI

Pai Rico, Pai Pobre - Edição de 20 anos atualizada e ampliada

T. Kiyosaki, Robert
9788550801483
336 páginas

[Compre agora e leia]

A escola prepara as crianças para o mundo real? Essa é a primeira pergunta com a qual o leitor se depara neste livro. O recado é ousado e direto: boa formação e notas altas não bastam para assegurar o sucesso de alguém. O mundo mudou; a maioria dos jovens tem cartão de crédito, antes mesmo de concluir os estudos, e nunca teve aula sobre dinheiro, investimentos, juros etc. Ou seja, eles vão para a escola, mas continuam financeiramente improficientes, despreparados para enfrentar um mundo que valoriza mais as despesas do que a poupança. Para o autor, o conselho mais perigoso que se pode dar a um jovem nos dias de hoje é: "Vá para a escola, tire notas altas e depoisprocure um trabalho seguro." O fato é que agora as regras são outras, e não existe mais emprego garantido para ninguém. Pai Rico, Pai Pobre demonstra que a questão não é ser empregado ou empregador, mas ter o controle do próprio destino ou delegá-lo a alguém. É essa a tese de Robert Kiyosaki neste livro substancial e visionário. Para ele, a formação proporcionada pelo sistema educacional não prepara os jovens para o mundo que encontrarão depois de formados. E como os pais podem ensinar aos filhos o que a escola relega? Essa é outra das muitas perguntas que o leitor encontra em Pai Rico, Pai Pobre. Nesse sentido, a proposta do autor é facilitar a tarefa dos pais. Quem entende de contabilidade deve esquecer seus conhecimentos acadêmicos, pois muitas das teorias expostas por Robert Kiyosaki contrariam os princípios contábeis comumente aceitos, e apresentam uma valiosa e moderna percepção do modo como se realizam os investimentos. A sociedade sofre mudanças radicais e, talvez, de proporções maiores do que as ocorridas em séculos passados. Não existe bola de cristal, mas algo é certo: a perspectiva global de transformações transcende nossa realidade imediata. Aconteça o que acontecer, só existem duas alternativas: segurança ou independência financeira. E o objetivo de Pai Rico, Pai Pobre é instruir o leitor e despertar sua inteligência financeira e a de seus filhos.

Best-seller internacional

JORDAN B. PETERSON

12 REGRAS PARA A VIDA

UM ANTÍDOTO PARA O CAOS

"Um dos pensadores mais importantes a surgir no cenário mundial em muitos anos." THE SPECTATOR

PREFÁCIO DE NORMAN DOIDGE

ALTA BOOKS
EDITORA

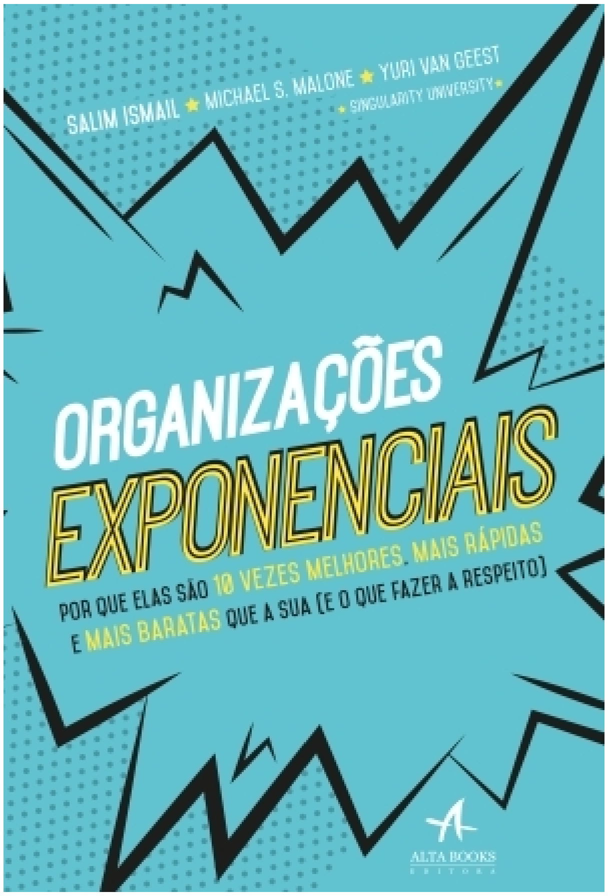
12 Regras para a Vida

Peterson, Jordan B.
9788550804002
448 páginas

[Compre agora e leia]

Aclamado psicólogo clínico, Jordan Peterson tem influenciado a compreensão moderna sobre a personalidade e, agora, se transformou em um dos pensadores públicos mais populares do mundo, com suas palestras sobre tópicos que variam da bíblia, às relações amorosas e à mitologia, atraindo dezenas de milhões de espectadores. Em uma era de mudanças sem precedentes e polarização da política, sua mensagem franca e revigorante sobre o valor da responsabilidade individual e da sabedoria ancestral tem ecoado em todos os cantos do mundo. Bem-humorado, surpreendente e informativo, dr. Peterson nos conta por que meninos e meninas andando de skate devem ser deixados em paz, que terrível destino aguarda aqueles que criticam com muita facilidade e por que você sempre deve acariciar gatos ao encontrar um na rua. O que o sistema nervoso das humildes lagostas tem a nos dizer sobre a relação entre manter as costas eretas (e os ombros para trás) e o sucesso na vida? Por que os antigos egípcios veneravam a capacidade de atenção como seu deus mais supremo? Que terríveis caminhos as pessoas percorrem quando se tornam ressentidas, arrogantes e vingativas? Neste livro, ele oferece doze princípios profundos e práticos sobre como viver uma vida com significado.

[Compre agora e leia]

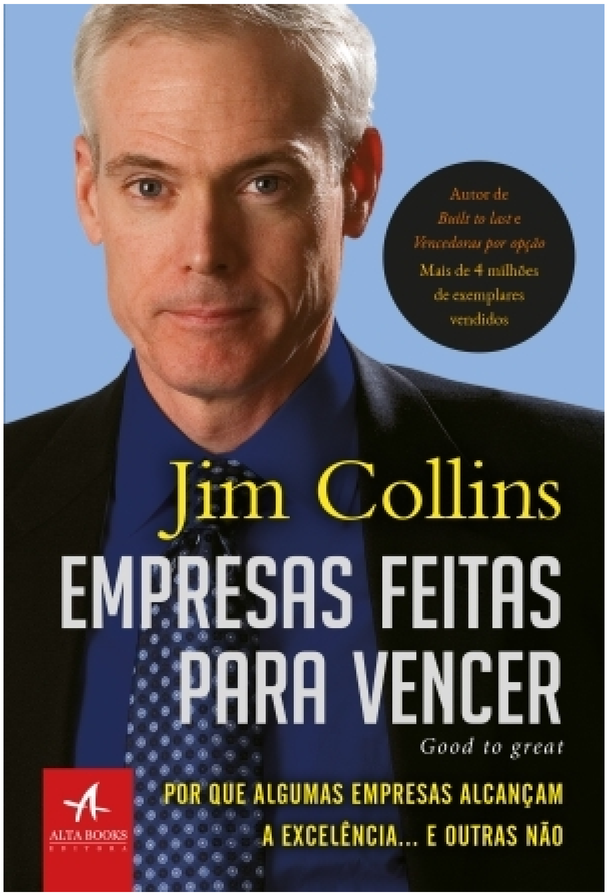

SALIM ISMAIL ★ MICHAEL S. MALONE ★ YURI VAN GEEST
★ SINGULARITY UNIVERSITY ★
ORGANIZAÇÕES
EXPONENCIAIS
POR QUE ELAS SÃO 10 VEZES MELHORES, MAIS RÁPIDAS
E MAIS BARATAS QUE A SUA (E O QUE FAZER A RESPEITO)
ALTA BOOKS
EDITORA

Organizações exponenciais

Ismail, Salim
9788550805108
288 páginas

[Compre agora e leia]

Bem-vindo à época das mudanças exponenciais, o melhor momento para se viver. É o momento em que a concorrência não é mais a empresa multinacional no exterior, mas o cara em uma garagem no Vale do Silício ou em Bandra (Mumbai), utilizando as mais recentes ferramentas online para projetar e imprimir a partir da nuvem sua última criação. Essas empresas estão cada vez mais rápidas, contam com pessoas cada vez mais capazes de se reinventar a uma velocidade ímpar. São pessoas e empresas extremamente criativas. Como aproveitar todo esse poder criativo? Como construir uma empresa que seja tão ágil, hábil e inovadora como as pessoas que farão parte dela? Como competir nesse acelerado mundo novo? Como se organizar para expandir? A resposta é a organização exponencial. Não temos escolha, porque em muitos setores a aceleração já está em andamento. Peter Diamandis usa o conceito dos 6 Ds: digitalizado, disfarçado, disruptivo, desmaterializar, desmonetizar e democratizar. Qualquer tecnologia que se torna digitalizada (primeiro "D") entra em um período de crescimento disfarçado. O que os autores têm observado – e que você perceberá lendo este livro – é que nenhuma empresa comercial, governamental ou sem fins lucrativos, conforme configurada no momento, pode acompanhar o ritmo que será definido por estes 6 Ds. Para isso será necessário algo radicalmente novo – uma nova visão da organização que seja tão tecnologicamente inteligente, adaptável e abrangente quanto o novo mundo em que vai operar – e, no final de tudo, transformar. Precisamos estar preparados para as mudanças. Sabe por quê? Porque, enquanto você leu este texto, muita coisa mudou. Boa leitura!


Autor de
Built to last e
Vencedoras por opção
Mais de 4 milhões
de exemplares
vendidos
Jim Collins
EMPRESAS FEITAS
PARA VENCER
Good to great
POR QUE ALGUMAS EMPRESAS ALCANÇAM
A EXCELÊNCIA... E OUTRAS NÃO
ALTA BOOKS

Empresas feitas para vencer

Collins, Jim
9788550804989
368 páginas

Compre agora e leia

Este clássico dos livros de negócios responde à seguinte pergunta: Como empresas boas, medianas e até ruins podem atingir uma qualidade duradoura? A obra de Jim Collins mostra como as grandes empresas triunfam no decorrer do tempo e como o desempenho sustentável a longo prazo pode ser inserido no DNA de uma organização desde sua concepção. Collins apresenta exemplos que desafiam a lógica e transformam a mediocridade em uma superioridade duradoura. O autor descreve também as características universais que levam uma empresa a se tornar excelente e outras não. Os resultados do estudo vão surpreender muitos leitores e lançar novas abordagens sobre quase todas as áreas da gestão. Depoimentos: "Um dos livros de negócios mais importantes de todos os tempos." Time Magazine. "Algumas de nossas decisões empresariais mais importantes foram tomadas com o apoio do raciocínio lógico de Collins." Jorge Paulo Lemann. "Jim Collins é o mais importnate influente pensador dos negócios da atualidade." Fortune. "Para Collins, nenhum problema é grande demais. The New York Times

Compre agora e leia