# Computação

## Noções de Lógica

Gustavo Augusto Lima de Campos
Jerffeson Teixeira de Souza

Ed UECE

Geografia

História

Educação Física

Química

Ciências Biológicas

Artes Plásticas

Computação

Física

Matemática

Pedagogia

# Computação

## Noções de Lógica

Gustavo Augusto Lima de Campos
Jerffeson Teixeira de Souza

3ª edição
Fortaleza - Ceará

Ed UECE

2015




Editora Filiada à

ASSOCIAÇÃO BRASILEIRA DAS EDITORAS UNIVERSITÁRIAS



Dados Internacionais de Catalogação na Publicação
Sistema de Bibliotecas
Luciana Oliveira – CRB-3 / 304
Bibliotecário

C198n Campos, Gustavo Algusto Lima de.
Noções de Lógica / Gustavo Augusto Lima de Campos , Jerfferson Teixeira de Souza. : – 3. ed. – Fortaleza : EdUECE, 015.
95 p. : il. ; 20,0cm x 25,5cm. (Computação)

Inclui bibliografia.

ISBN: 978-85-7826-445-1

1. Computação. 2. Lógica – Computação. I. Souza, Jerffeson II. Título.

CDD 511.3

Editora da Universidade Estadual do Ceará – EdUECE
Av. Dr. Silas Munguba, 1700 – Campus do Itaperi – Reitoria – Fortaleza – Ceará
CEP: 60714-903 – Fone: (85) 3101-9893
Internet: www.uece.br – E-mail: eduece@uece.br

Secretaria de Apoio às Tecnologias Educacionais
Fone: (85) 3101-9962

# Sumário

# Apresentação

Este livro destaca algumas das principais noções presentes no estudo da ciência do Raciocínio Lógico. Primeiramente, o livro enfatiza o uso da linguagem Lógica na representação do conhecimento e os princípios que são empregados na demonstração da validade de argumentos. Posteriormente, enfatiza a automação dos processos envolvidos na demonstração de validade e sua utilização no contexto da programação em lógica. O conteúdo do livro foi divido em três unidades: Lógica Proposicional, Lógica de Predicados e Resolução.

A Parte 1 apresenta informalmente os conceitos de proposições e proposições compostas, de teoria e raciocínio, e de sistemas formais. Em seguida, apresenta o sistema formal Lógica Proposicional em duas partes. A primeira parte apresenta a linguagem formal lógica proposicional, a semântica dos conectivos lógicos e as tabelas verdade, as noções de equivalência lógica e implicação lógica, de tautologia e contradição. A segunda parte apresenta a noção de argumento e o processo de demonstração de validade de um argumento.

A Parte 2 apresenta a linguagem Lógica de Predicados, a geração de fórmulas bem formadas na linguagem e a semântica de proposições envolvendo quantificadores. Essa Unidade enfatiza a representação de proposições em Lógica de Predicados e identifica a analogia entre o processo de demonstração de validade de argumentos e a noção de programação em lógica, onde o conceito de computação se confunde com o conceito de dedução, que, na Unidade, é exemplificada com o método do raciocínio para trás a partir do objetivo a ser demonstrado, ou seja, de uma consulta a ser respondida por um programa em lógica.

A Parte 3 apresenta idéias e algoritmos associados à prova automática de argumentos empregando o método da Resolução para a Lógica de Predicados. São detalhados o algoritmo da Resolução e os algoritmos de Conversão para a Forma Clausal, que é o tipo de fórmula bem formada manipulada pela Resolução, e o algoritmo da Unificação, que é necessário durante o processo de resolução e para a obtenção de respostas para consultas envolvendo variáveis.

**Os Autores**

# Parte 1

# Lógica Proposicional

# Capítulo 1

# Introdução à Lógica

## Objetivos

- Conhecer a linguagem formal Lógica Proposicional, assim como gerar fórmulas bem formadas nessa linguagem e atribuir valor verdade às fórmulas, envolvendo conectivos lógicos, as regras de inferência que são empregadas no raciocínio correto e a forma geral em que os argumentos são estabelecidos.
- Aplicar o conhecimento adquirido nos processos de representação de teorias e argumentos em Lógica Proposicional e de demonstração de validade de argumentos.

## Introdução

O objetivo deste capítulo consiste em introduzir informalmente a idéia do que vem a ser Lógica e alguns outros conceitos básicos, que acreditamos facilitarão a compreensão dos objetivos da Disciplina e dos assuntos abordados nos próximos capítulos e unidades.

De uma maneira geral, as apologias abaixo definem informalmente o que vem a ser Lógica:

> Lógica é a ciência do raciocínio. (Malba Tahan)
>
> Lógica é a ciência das leis do pensamento e a arte de aplicá-las corretamente na pesquisa e na demonstração da verdade. (R. Solivete)
>
> A Lógica é a ciência que dirige, por meio de leis, as operações de nossa razão, para que ordenada, facilmente alcance a verdade. (Sinibaldi)

Por sua vez, podemos entender o ato de raciocinar como um processo de derivação de novos conhecimentos a partir de conhecimentos antigos. Em nosso curso, o conhecimento é expresso através de um conjunto de proposições. Uma **proposição** é uma sentença declarativa à qual podemos atribuir

um dos valores verdade: Verdadeiro (V) ou Falso (F). A proposição é o bloco construtor da Lógica. Por exemplo, as seguintes sentenças declarativas são **proposições simples:**

- "Está chovendo"
- "2 é maior que 3"
- "3 é menor que 4"

Neste exemplo, podemos afirmar que, dependendo das condições climáticas em um dado momento, a primeira proposição pode ser V ou F, a segunda é uma proposição F e a terceira é V.

Por outro lado, existem algumas sentenças declarativas às quais não conseguimos atribuir um valor verdade. Por exemplo, as sentenças abaixo não são proposições:

- "x é menor que 100"
- "Esta sentença é falsa"

Neste caso, a não ser que saibamos o valor de x, no primeiro exemplo, e a sentença que estamos afirmando ser falsa, no segundo, não conseguimos atribuir um valor verdade para as sentenças.

Além das proposições simples, estamos acostumados a construir **proposições compostas** a partir da combinação de proposições simples (subproposições) e conectivos (*e, ou, não, se-então, se e somente se*). Por exemplo, as sentenças abaixo são proposições compostas:

- "Está chovendo e 2 é maior que 3"
- "2 é maior que 3 ou 3 é menor que 4"

Neste caso, independentemente das condições climáticas, a primeira proposição é F e a segunda é V (se você tem dúvida, aguarde até a apresentação das tabelas verdade dos conectivos *e* e *ou*).

Uma *teoria* consiste de um conjunto de proposições a respeito de um mundo particular. Por exemplo, o conjunto formado pelas três proposições que descrevem o estado de espírito de Sócrates e Platão pode ser visto como uma teoria:

- "Se Platão estiver disposto a visitar Sócrates então Sócrates está disposto a visitar Platão."
- "Se Sócrates estiver disposto a visitar Platão então Platão não está disposto a visitar Sócrates."
- "Se Sócrates não estiver disposto a visitar Platão então Platão está disposto a visitar Sócrates."

Um **sistema formal** consiste de uma linguagem formal, apropriada para representar teorias, e de uma abstração adequada para os princípios usados

para provar quando certas proposições são conseqüências lógicas de proposições que compõem teorias.

A **Lógica Proposicional** é um sistema formal apropriado para a representação de teorias simples, ou seja, aquelas compostas de proposições elementares (proposições que não envolvem quantificadores e variáveis). Veja como a teoria que fala a respeito do estado de espírito de Sócrates e Platão poderia ser representada utilizando-se uma Linguagem Lógica Proposicional:

- $p \rightarrow q$
- $q \rightarrow \neg p$
- $\neg q \rightarrow p$

Representando a Teoria sobre o Mundo de Sócrates e Platão através da linguagem formal acima, podemos fazer uso de vários mecanismos disponíveis para a obtenção de conseqüências lógicas de proposições conhecidas. Por exemplo, utilizando uma Tabela Verdade podemos concluir que Sócrates está disposto a visitar Platão, ou seja:

| p | q | ¬p | ¬q | p → q | q → ¬ p | ¬q → p | (p → q) ∧ (q → ¬p) ∧ (¬q → p) | ((p → q) ∧ (q → ¬p) ∧ (¬q → p)) → q |
|---|---|---|---|---|---|---|---|---|
| F | F | V | V | V | V | F | F | V |
| F | V | V | F | V | V | V | V | V |
| V | F | F | V | F | V | V | F | V |
| V | V | F | F | V | F | V | F | V |

Nos próximos capítulos e unidades, desejamos apresentar as linguagens formais Lógica Proposicional e Lógica de Predicados, e os diversos mecanismos de inferência disponíveis para a obtenção de conseqüências lógicas.

Capítulo 2

# Lógica Proposicional

## 1. Definição de uma Linguagem Proposicional

A definição do sistema formal Lógica Proposicional passa pela definição da **Linguagem Proposicional** e dos princípios que governam os conectivos lógicos pertencentes ao seu alfabeto. Por sua vez, a definição desta linguagem passa pela definição de um **alfabeto** de símbolos, empregados na construção de fórmulas, e das **regras sintáticas** para a geração de fórmulas bem formadas (*fbfs)*.

O alfabeto da Linguagem Proposicional é definido a partir do seguinte conjunto de símbolos:

- conjunto de símbolos proposicionais: p, q, r, ...;
- conectivos lógicos: $\wedge$, $\vee$, $\neg$, $\rightarrow$, $\leftrightarrow$;
- parênteses: ( , ) .

Os símbolos proposicionais são empregados na representação das subproposições, ou seja, das proposições simples estabelecidas em linguagem natural (Português, Inglês, etc.). Os conectivos lógicos representam respectivamente as partículas *e, ou, não, se-então* e *se e somente se*. Os parênteses servem para denotar pontuação. Por exemplo, tente perceber a convenção geralmente adotada quando da utilização de parênteses na representação de proposições escritas em linguagem natural:

*"3+1 = 5 ou 1=1, implica 3 = 3"* representada por $(p \vee q) \rightarrow r$

*"3+1 = 5 ou 1=1 implica 3 = 3"* representada por $p \vee (q \rightarrow r)$

De acordo com a convenção acima, a primeira e a segunda proposição também podem ser expressas respectivamente das seguintes maneiras: "se 3+1 = 5 ou 1=1, então 3 = 3" e "3+1 = 5 ou se 1=1 então 3 = 3".

As regras sintáticas da Linguagem Proposicional definem o conjunto de fórmulas bem formadas (fbfs) na Linguagem como sendo:

- os símbolos proposicionais são fbfs: p, q, r, ...;
- se p e q são fbfs então $p \wedge q$, $p \vee q$, $\neg p$, $p \rightarrow q$, $p \leftrightarrow q$ são fbfs;
- se p e q são fbfs então $(p \wedge q)$, $(p \vee q)$, $\neg(p)$, $(p \rightarrow q)$, $(p \leftrightarrow q)$ são fbfs.

A Linguagem Proposicional pode ser definida como o conjunto de todas as fbfs possíveis de serem geradas a partir do alfabeto de símbolos e das regras sintáticas descritas acima.

## 2. E, Ou, Não e Tabelas Verdade

Nesta seção, primeiramente, desejamos criar proposições compostas em linguagem natural utilizando as partículas e, ou e não, e representar estas proposições como fbfs em linguagem Lógica Proposicional. De posse de uma fbf particular em Lógica Proposicional, as regras semânticas da linguagem capturam o significado pretendido dos conectivos, associando a cada fórmula um dos valores verdade: V ou F. Posteriormente, utilizamos as Tabelas Verdade como um mecanismo apropriado para o estudo dos significados destas fbfs. A seção foi dividida em quatro subseções principais:

1. Conectivo E
2. Conectivo OU
3. Conectivo NÃO
4. Tabelas Verdade

### 2.1 Conectivo E

Se p *e* q são duas proposições então "p *e* q" é também uma proposição. Podemos dizer que:

- *"p e q"* é denominada conjunção de p e q;
- *"p e q"* é representada por $p \wedge q$.

De acordo com o esquema de representação adotado acima, dizemos que:

- se ambas as proposições, p e q, são verdadeiras,

  então $p \wedge q$ é verdadeira

  senão $p \wedge q$ é falsa.

Assim, de acordo com a proposição acima, o significado de $p \wedge q$ pode ser expresso através da seguinte tabela verdade:

| p | q | $p \wedge q$ |
|---|---|---|
| F | F | F |
| F | V | F |
| V | F | F |
| V | V | V |

Por exemplo, as três primeiras proposições abaixo são falsas (F) e a última é verdadeira (V):

"3+1=6 e 2+2=5"

"2=5 e 2=2"

"2=2 e 2=3"

"2=2 e 3+4=7"

## 2.2 Conectivo Ou

Se p e q são duas proposições então "p *ou* q" é também uma proposição. Podemos dizer que:

- *"p ou q"* é denominada disjunção de p e q;
- *"p ou q"* é representada por $p \vee q$.

De acordo com o esquema de representação adotado acima, dizemos que:

- se **pelo menos** uma das proposições, p ou q, é verdadeira

  então $p \vee q$ é verdadeira

  senão $p \vee q$ é falsa.

Além do mais, o significado de $p \vee q$ pode ser expresso através da seguinte tabela verdade:

| p | q | $p \vee q$ |
|---|---|---|
| F | F | F |
| F | V | V |
| V | F | V |
| V | V | V |

Por exemplo, a primeira proposição abaixo é falsa e as três últimas são verdadeiras:

"2 = 3 ou 2 + 2 = 5"

"2 = 3 ou 3 = 3"

"2 = 2 ou 2 = 3

"2 = 2 ou 3 + 4 = 7"

Esta disjunção também é denominada "ou inclusivo", ou seja, corresponde ao ***e/ou*** algumas vezes encontrada em documentos legais. Neste caso, conforme você pode observar na tabela verdade que a define, a proposição composta é verdadeira, inclusive, quando ambas as subproposições envolvidas são verdadeiras. Na conversação ordinária freqüentemente usamos ou no sentido exclusivo, por exemplo:

*"Quando você telefonou eu estava tomando banho ou estava pas--seando"*

Neste caso, a verdade da proposição acima não incluí as duas subproposições, ou seja, ela é verdadeira quando **exatamente uma** das subproposições é verdadeira.

Considerando a observação acima, podemos utilizar a seguinte tabela verdade para definir o ou-excluivo:

| p | q | p⊕q |
|---|---|---|
| F | F | F |
| F | V | V |
| V | F | V |
| V | V | F |

Observe que, apesar de estarmos apresentando o ou-exclusivo, o símbolo $\oplus$ não pertence ao conjunto de símbolos que define o alfabeto da Linguagem Proposicional. Mais adiante, você poderá observar que este símbolo não precisa fazer parte do alfabeto de símbolos da Linguagem Proposicional, já que é possível construir seu significado, ou melhor, sua a tabela verdade, a partir de pelo menos dois dos conectivos componentes do alfabeto.

## 2.3 Conectivo Não

Se p é uma proposição então "não p" é também uma proposição. Podemos dizer que:

- "não p" é denominada negação de p;
- "não p" é representada por ¬p.

De acordo com o esquema de representação adotado acima, dizemos que:

- se uma proposição, p, é verdadeira

  então ¬p é falsa

  senão ¬p é verdadeira.

Além do mais, o significado de ¬p pode ser expresso através da seguinte tabela verdade:

| p | ¬p |
|---|---|
| F | V |
| V | F |

Observe que existem várias maneiras de negar uma proposição escrita em linguagem natural. Por exemplo, considere as proposições abaixo:

**1.** "2 + 2 = 5"

**2.** "Não é o caso que 2 + 2 = 5"

**3.** "$2 + 2 \neq 5$"

**4.** "$2 + 2 > 5$"

**5.** "$2 + 2 \leq 5$"

**6.** "$x^2 + 5x - 1$ não é uma equação quadrática"

**7.** "Não é verdade que $x^2 + 5x - 1$ não é uma equação quadrática"

**8.** "$x^2 + 5x - 1$ é uma equação quadrática"

**Note que:**

- a segunda e a terceira proposições são negações da primeira;
- a quinta proposição é a negação da quarta;
- a sétima e oitava proposições são negações da sexta.

Nesse curso de Noções de Lógica estaremos considerando que o símbolo ¬ se aplica somente ao próximo símbolo proposicional, ou seja:

$\neg p \vee q$ significa $\neg(p) \vee q$

$\neg p \vee q$ não significa $\neg(p \vee q)$

Além do mais, considerando a representação de proposições em linguagem natural, adotaremos a seguinte convenção:

$\neg p \vee q$ representa *"Não é o caso que p, ou q"*

$\neg(p \vee q)$ representa *"Não é o caso que p ou q"*

## 2.4 Tabelas Verdade

Como você já deve ter percebido, as tabelas verdade podem ser usadas para expressar os valores verdade possíveis de proposições compostas. A construção das várias colunas de uma tabela verdade pode ser realizada de uma maneira sistemática. Por exemplo, observe a construção da tabela verdade de $\neg(p \vee \neg q)$:

**Passo 1:** preencher os valores verdade possíveis de p e q

| p | q | ¬(p ∨ ¬q) |
|---|---|---|
| F | F | |
| F | V | |
| V | F | |
| V | V | |

**Passo 2:** preencher coluna ¬q

| p | q | ¬q | ¬(p ∨ ¬q) |
|---|---|---|---|
| F | F | V | |
| F | V | F | |
| V | F | F | |
| V | V | V | |

**Passo 3:** preencher coluna p ∨ ¬q

| p | q | ¬q | (p ∨ ¬q) | ¬(p ∨ ¬q) |
|---|---|---|---|---|
| F | F | V | V | |
| F | V | F | F | |
| V | F | V | V | |
| V | V | F | V | |

**Passo 4:** preencher coluna ¬(p ∨ ¬q)

| p | q | ¬q | (p ∨ ¬q) | ¬(p ∨ ¬q) |
|---|---|---|---|---|
| F | F | V | V | F |
| F | V | F | F | V |
| V | F | V | V | F |
| V | V | F | V | F |

Após um período de experiência, alguns dos passos escritos acima podem ser eliminados. Observe que se uma proposição composta envolve n subproposições então sua tabela verdade tem 2n linhas. Por exemplo, uma proposição composta por 3 subproposições tem 23 (oito) linhas.

### Para refletir

**1.** Atribua valores verdade para as seguintes proposições:

a) 3 ≤ 7 e 4 é um inteiro impar.

b) 3 ≤ 7 ou 4 é um inteiro impar.

c) 2 + 1 = 3 mas 4 < 4.

d) 5 é impar ou divisível por 4.

e) Não é verdade que 2 + 2 = 5 e 5 > 7.

f) Não é verdade que 2 + 2 = 5 ou 5 > 7.

g) 3 ≥ 3.

**2.** Suponha que p represente a proposição "7 é um inteiro par", q represente "3 + 1 = 4" e r represente "24 é divisível por 8".

a) Escreva as seguintes proposições em formas simbólicas e atribua valores verdade:

i) 3 +1 ≠ 4 e 24 é divisível por 8.

ii) Não é verdade que 7 é impar ou 3 + 1 = 4.

iii) 3 + 1 = 4 mas 24 não é divisível por 8.

b) Escreva as seguintes formas simbólicas em palavras e atribua valores verdade:

i) $p \vee \neg q$.

ii) $\neg(r \wedge q)$.

iii) $\neg r \vee \neg q$.

**3.** Construa tabelas verdade para

a) $\neg p \vee q$.

b) $\neg p \wedge p$.

c) $(\neg p \vee q) \wedge r$.

d) $\neg(p \wedge q)$.

e) $\neg p \wedge \neg q$.

f) $\neg p \vee \neg q$.

g) $p \vee \neg p$.

h) $\neg(\neg p)$.

**4.** Apresente negações adequadas para

a) 3 - 4 < 7.

b) 3 + 1 = 5 e 2 ≤ 4.

c) 8 é divisível por 3 mas 4 não é.

**5.** Suponha que definamos o conectivo © da seguinte maneira: se p é falsa e q é verdadeira então p © q é verdadeira, senão p © q é falsa.

a) Escreva a tabela verdade para p © q.

b) Escreva a tabela verdade para q © p.

c) Escreva a tabela verdade para (p © p) © q.

**6.** Denotemos o "ou exclusivo", algumas vezes utilizado em nossas conversações ordinárias, por $\oplus$. Na definição de p $\oplus$ q, se exatamente uma das formas p, q é verdadeira então p $\oplus$ q é verdadeira, senão p $\oplus$ q é falsa.

a) Escreva a tabela verdade para p $\oplus$ q.

b) Escreva as tabelas verdade para p $\oplus$ p e (p $\oplus$ q) $\oplus$ q.

c) Mostre que "e/ou" realmente significa "e ou ou", isto é, que a tabela verdade para $p \vee q$ é a mesma que $(p \wedge q) \oplus (p \oplus q)$.

d) Mostre que as formas $(p \wedge q) \vee (p \vee q)$, $(p \vee q) \oplus (p \oplus q)$ possuem o mesmo significado, ou seja, "e ou ou" pode ser representado tanto por $(p \wedge q) \vee (p \vee q)$ quanto por $(p \wedge q) \oplus (p \oplus q)$.

## 3. Implicação e o Bi-condicional

Uma das mais importantes formas matemáticas é a implicação. A maioria dos teoremas matemáticos é descrita neste formato, ou seja, como uma proposição do tipo "se hipótese então conclusão". O Bi-condicional é uma forma matemática que pode ser composta a partir de duas implicações e uma conjunção. Podermos demonstrar esta afirmação a partir do momento que apresentarmos uma definição para o conceito de equivalência lógica. Dividimos esta seção em três subseções principais:

1. Equivalência Lógica
2. Implicação
3. Bi-condicional

### 3.1 Equivalência Lógica

Se duas proposições p, q têm a mesma tabela verdade então p é logicamente equivalente a q. Podemos dizer que:

p é **logicamente equivalente** a q é representada por $p \Leftrightarrow q$.

Quando duas proposições são logicamente equivalentes, elas têm a **mesma forma e**, **conseqüentemente, podemos substituir uma pela outra em qualquer proposição ou teorema** (aguarde um pouco mais e você poderá verificar o que estamos mencionando). Veja a equivalência lógica através da construção das tabelas verdade das proposições $\neg (p \wedge q)$ e $\neg p \vee \neg q$:

| p | q | $\neg p$ | $\neg q$ | $(p \wedge q)$ | $\neg(p \wedge q)$ | $\neg p \vee \neg q$ |
|---|---|---|---|---|---|---|
| F | F | V | V | F | V | V |
| F | V | V | F | F | V | V |
| V | F | F | V | F | V | V |
| V | V | F | F | V | F | F |

Observe que, independentemente de sabermos o que p e q representam, podemos afirmar que a fbf $\neg (p \wedge q)$ é logicamente equivalente a fbf $\neg p \vee \neg q$. É importante ressaltar que a forma de uma proposição é que deter-

mina se a ela é (ou se ela não é) logicamente equivalente a uma outra proposição, e não o valor verdade das proposições envolvidas.

Por exemplo, as proposições "2 + 5 = 7" e "3 - 1 = 2" são proposições verdadeiras mas elas não são logicamente equivalentes. Para comprovar, represente a primeira proposição pelo símbolo proposicional p e a segunda por q e, em seguida, verifique as tabelas verdade das duas fbfs.

Por outro lado, "2 + 5 = 7 ou 3 - 1 = 2" e "3 - 1 = 2 ou 2 + 5 = 7", além de serem proposições verdadeiras, são logicamente equivalentes. Para comprovar, considerando o esquema de representação adotado no parágrafo anterior, represente a primeira proposição por $p \vee q$ e a segunda por $q \vee p$ e, em seguida, verifique se as formas têm tabelas verdade idênticas.

As **Leis de DeMorgan** utilizam a ideia de equivalência lógica para estabelecer a relação existente entre a negação, a conjunção e a disjunção:

- se p e q são proposições então:

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

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

Em palavras, as Leis de DeMorgan estabelecem que a negação de uma disjunção é logicamente equivalente a conjunção de negações, e que a negação de uma conjunção é logicamente equivalente a disjunção de negações.

## 3.2 Implicação

Se p, q são proposições então "se p então q" é também uma proposição. Podemos dizer que:

- "se p então q" é denominada condicional entre p e q;
- "se p então q" é representada por $p \rightarrow q$.

Nesta proposição, p é denominada premissa (ou hipótese ou antecedente) e q é denominada conclusão (ou conseqüência ou conseqüente).

De acordo com o esquema de representação adotado acima, dizemos que:

- se p é uma proposição verdadeira e q é uma proposição falsa

  então $p \rightarrow q$ é falsa

  senão $p \rightarrow q$ é verdadeira.

Além do mais, o significado de $p \rightarrow q$ pode ser expresso através da seguinte tabela verdade:

| p | q | $p \rightarrow q$ |
|---|---|---|
| F | F | V |
| F | V | V |
| V | F | F |
| V | V | V |

Por exemplo, a primeira, a segunda e a quarta proposições abaixo são verdadeiras e a terceira é falsa:

*"Se verde é vermelho então a lua é feita de queijo"*

*"Se verde é vermelho então 2 = 2"*

*"Se 2 = 2 então verde é vermelho"*

*"Se 2 = 2 então a lua não é feita de queijo"*

Podemos compreender melhor a tabela verdade da implicação, buscando um esclarecimento para o significado de uma proposição do tipo se p então q. Uma proposição deste tipo nos diz exatamente que se p, o antecedente, ocorrer (verdadeiro) então q, o conseqüente, também deve ocorrer. Neste caso, dizemos que a ocorrência de p é **suficiente** para garantir a ocorrência de q, e, também, que quando p ocorrer q deve **necessariamente** ocorrer.

Considerando este significado, podemos justificar a quarta linha da tabela verdade do condicional, ou seja, na situação em que o antecedente e o conseqüente são *V*, o condicional é, também, *V*, já que o significado da proposição se **antecedente** então **consequente** está sendo respeitado.

Seguindo esta linha de raciocínio, somente a terceira linha da tabela parece violar o significado deste tipo de proposição, já que, nesta linha, o antecedente ocorre e o conseqüente não ocorre (falso), ou seja, quando o antecedente é *V* e o conseqüente é *F*, o condicional das subproposições é falso.

Além do mais, observe que a proposição se **antecedente** então **consequente** não nos diz nada a respeito de qual deve ser o estado do conseqüente (*V* ou *F*?) quando o antecedente é *F*. Isto significa que tanto a primeira linha da tabela (antecedente *F* e conseqüente *F*) quanto a segunda (antecedente *F* e conseqüente *V*) não violam o significado da proposição se-então, ou seja, o condicional de subproposições cujos valores verdade são semelhantes aos destas linhas é *V*.

Uma melhor compreensão do significado de uma proposição no formato de um condicional, também, dá origem às seguintes equivalência lógicas:

- $p \rightarrow q \Leftrightarrow \neg q \rightarrow \neg p$
- $p \wedge \neg q \Leftrightarrow \neg (p \rightarrow q)$

Ou seja: quando $p \rightarrow q$ é verdadeira então $\neg q \rightarrow \neg p$ é verdadeira e vice-versa; e quando $p \wedge \neg q$ é verdadeira então é porque $\neg (p \rightarrow q)$ é verdadeira (isto é, $p \rightarrow q$ é falsa) e vice-versa. A proposição $\neg q \rightarrow \neg p$ é denominada **contrapositiva** de $p \rightarrow q$.

Além do formato se p então q, existem outras maneiras de se estabelecer o condicional em português:

*"Se p então q"*

*"p implica q"*

*"p é mais forte que q"*

*"q é mais fraca que q"*

*"p somente se q*

*"q se p"*

*"p é suficiente para q"*

*"q é necessária para p"*

*"Uma condição necessária para p é q"*

*"Uma condição suficiente para q é p"*

## 3.3 Bi-condicional

Se p, q são proposições então "p se e somente se q" (algumas vezes abreviado sss) é também uma proposição. Podemos dizer que:

- *"p se e somente se q"* é denominada bi-condicional entre p e q;
- *"p se e somente se q"* é representada por $p \leftrightarrow q$.

De acordo com o esquema de representação adotado acima, dizemos que:

- se p e q têm o mesmo valor verdade

  então $p \leftrightarrow q$ é verdadeira

  senão $p \leftrightarrow q$ é falsa.

Além do mais, o significado de $p \leftrightarrow q$ pode ser expresso através da seguinte tabela verdade:

| p | q | $p \leftrightarrow q$ |
|---|---|---|
| F | F | V |
| F | V | F |
| V | F | F |
| V | V | V |

Por exemplo, a primeira e a quarta proposições abaixo são verdadeiras, e a segunda e a terceira são falsas:

*"verde é vermelho se e somente se a lua é feita de queijo"*

*"verde é vermelho se e somente se 2 = 2"*

*"2 = 2 se e somente se verde é vermelho*

*"2 = 2 se e somente se a lua não é feita de queijo"*

Existem outras maneira de se estabelecer o bi-condicional em português:

- "p é necessária e suficiente para q"
- "p é equivalente a q"

De acordo com o que o nome bi-condicional indica, existe uma conexão entre este conectivo e o condicional. Por exemplo, observe que "p se e somente se q" significa que $q \rightarrow p$ e $\neg q \rightarrow \neg p$, ou seja:

$$p \leftrightarrow q \Leftrightarrow (q \rightarrow p) \wedge (\neg q \rightarrow \neg p).$$

Além do mais, de acordo com a subseção anterior, $p \leftrightarrow q \Leftrightarrow \neg q \rightarrow \neg p$. Assim, a relação entre o bi-condicional e o condicional pode, também, ser expressa, de uma maneira mais simples, através da seguinte equivalência lógica:

$$p \leftrightarrow q \Leftrightarrow (p \rightarrow q) \wedge (q \rightarrow p).$$

Ou seja, quando $p \leftrightarrow q$ é verdadeira então $p \rightarrow q$ é verdadeira e $q \rightarrow p$ é verdadeira, e vice-versa.

## Para refletir

**1.** Quais das formas abaixo são logicamente equivalentes?

a) $p \wedge \neg q$.
b) $p \rightarrow q$.
c) $\neg (\neg p \vee q)$.
d) $q \rightarrow \neg p$.
e) $\neg p \vee q$.
f) $\neg (p \rightarrow q)$.
g) $p \rightarrow \neg q$.
h) $\neg p \rightarrow \neg q$.

**2.** Mostre que os seguintes pares (formas) são logicamente equivalentes:

a) $p \wedge (q \vee r)$; $(p \wedge q) \vee (p \wedge r)$.
b) $p \vee (q \wedge r)$; $(p \vee q) \wedge (p \vee r)$.
c) $p \leftrightarrow q$; $(p \rightarrow q) \wedge (q \rightarrow p)$.
d) $p \rightarrow q$; $\neg q \rightarrow \neg p$.

**3.** Mostre que os seguintes pares (formas) não são logicamente equivalentes:

a) $\neg (p \wedge q)$; $\neg p \wedge \neg q$.
b) $\neg (p \vee q)$; $\neg p \vee \neg q$.
c) $p \rightarrow q$; $q \rightarrow p$.
d) $\neg (p \rightarrow q)$; $\neg p \rightarrow \neg q$.

**4.** Indique quais proposições são verdadeiras:

a) Se 2 + 1 = 4 então 3 + 2 =5.

b) Vermelho é branco se e somente se verde é azul.

c) 2 + 1 = 3 e 3 + 1 = 5 implica 4 é impar.

d) Se 4 é impar então 5 é impar.

e) Se 4 é impar então 5 é par.

f) Se 5 é impar então 4 é impar.

**5.** Dê exemplos de proposições ou fale porque o exemplo não existe:

a) Uma implicação verdadeira com uma conclusão falsa.

b) Uma implicação verdadeira com uma conclusão verdadeira.

c) Uma implicação falsa com uma conclusão verdadeira.

d) Uma implicação falsa com uma conclusão falsa.

e) Uma implicação falsa com uma hipótese falsa.

f) Uma implicação falsa com uma hipótese verdadeira.

g) Uma implicação verdadeira com uma hipótese verdadeira.

h) Uma implicação verdadeira com uma hipótese falsa.

**6.** Transforme em símbolos:

a) *p sempre que q.*

b) *p a menos que q.*

**7.** Dê uma negação para $p \leftrightarrow q$ em uma forma que não envolva o bi-condicional.

**8.** Suponha que p, $\neg q$ e *r* são verdadeiras. Quais formas possuem interpretações verdadeiras?

a) $p \rightarrow q$.

b) $q \rightarrow p$.

c) $p \rightarrow (q \vee r)$.

d) $p \leftrightarrow q$.

e) $p \leftrightarrow r$.

f) $(p \vee q) \rightarrow p$.

g) $(p \wedge q) \rightarrow q$.

**9.** Temos cinco "conectivos" lógicos: $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$ e $\neg$, cada um corresponde à uma construção de nossa linguagem ordinária. Do ponto de vista lógico, poderíamos expressar todos estes conectivos em termos de (somente) $\neg$ e $\wedge$. Mais ainda, se definirmos p⁄q como sendo falsa quando tanto p e q são verdadeiras e p⁄q como sendo verdadeira em qualquer outro caso, poderíamos expressar todas as cinco formas em termos deste único conectivo. Verifique parcialmente as declarações dadas acima:

a) Achando uma proposição que é equivalente a $p \vee q$ usando somente $\wedge$ e $\neg$.

b) Escrevendo a tabela verdade para p⁄q.

c) Mostrando que p⁄p é equivalente a $\neg p$.

d) Mostrando que (p⁄q)⁄(q⁄p) é equivalente a $p \wedge q$.

## 4. Tautologias

As tautologias formam uma classe de proposições muito importante. São proposições compostas sempre verdadeiras, isto é, suas tabelas verdade contêm somente valores verdadeiros (Vs) na coluna final. O fato de uma proposição ser uma tautologia depende do formato da proposição, ou seja, da ordem em que os símbolos proposicionais são combinados com os conectivos e com os parênteses

para a formação da fbf (representação da proposição que estamos considerando ser uma tautologia). Por exemplo, a fbf $p \rightarrow (p \vee q)$ é uma tautologia:

| p | q | $(p \vee q)$ | $p \rightarrow (p \vee q)$ |
|---|---|---|---|
| F | F | F | V |
| F | V | V | V |
| V | F | V | V |
| V | V | V | V |

Observe na tabela verdade anterior, o fato de $p \rightarrow (p \vee q)$ ser uma tautologia independe dos significados atribuídos às subproposições envolvidas (dos significados de p e q), ou seja, toda proposição composta possível de ser representada por esta fbf configura uma tautologia. Por exemplo, considerando que p e q representem respectivamente *"bananas são laranjas"* e *"bananas são bananas"* (ou que p e q representem quaisquer outras duas proposições), o valor verdade de $p \rightarrow (p \vee q)$ é V, ou seja, "se bananas são laranjas então bananas são laranjas ou bananas são bananas" é uma proposição verdadeira (assim como, "se 2 = 2 então 2 = 2 ou 3 + 1 = 5" é, também, uma proposição verdadeira).

É importante que façamos a distinção entre proposições verdadeiras e tautologias. Nem sempre uma proposição verdadeira é uma tautologia. Por exemplo, "2 + 2 = 4" é uma proposição verdadeira mas não é uma tautologia pois, considerando sua representação por meio do símbolo proposicional p, a tabela verdade desta fbf nem sempre é verdadeira:

| p |
|---|
| F |
| V |

Por outro lado, para reforçar a ideia de uma tautologia, podemos dizer que a proposição "5 é a raiz primitiva de 17 ou 5 não é a raiz primitiva de 17" é uma tautologia, independentemente do que venha a ser a definição de raiz primitiva. Por exemplo, representando "5 é a raiz primitiva de 17" pelo símbolo proposicional p, observe que a tabela verdade da fbf $p \vee \neg p$ contém somente valores verdadeiros:

| p | q | $\neg p$ | $p \vee \neg p$ |
|---|---|---|---|
| F | F | V | V |
| F | V | V | V |
| V | F | F | V |
| V | V | F | V |

A negação de uma tautologia, isto é, uma proposição cuja tabela verdade contém somente valores falsos, é denominada contradição. Por exemplo, a fbf $(p \rightarrow q) \wedge (p \wedge \neg q)$ configura uma contradição:

| p | q | ¬ q | (p → q) | (p ∧ ¬ q) | (p → q) ∧ (p ∧ ¬ q) |
|---|---|---|---|---|---|
| F | F | V | V | F | F |
| F | V | F | V | F | F |
| V | F | V | F | V | F |
| V | V | F | V | F | F |

Observe que, como as tautologias, o fato de $(p \rightarrow q) \wedge (p \wedge \neg q)$ ser uma contradição independe dos significados atribuídos às subproposições envolvidas, ou seja, toda proposição composta possível de ser representada pela fbf $(p \rightarrow q) \wedge (p \wedge \neg q)$ configura uma contradição.

Também, é importante que façamos a distinção entre proposições falsas e contradições. Nem sempre uma proposição falsa é uma contradição. Por exemplo, "2 + 2 = 5" é uma proposição falsa mas não é uma tautologia pois, considerando que ela pode ser representada pelo símbolo proposicional q, sua tabela verdade nem sempre é falsa:

| q |
|---|
| F |
| V |

Por outro lado, "2 + 2 = 5 e 2 + 2 ≠ 5" é uma contradição. Por exemplo, considerando que esta proposição pode ser representada pela fbf $p \wedge \neg p$, observe que suaa tabela verdade contém somente valores falsos:

| q | ¬ q | q ∧ ¬ q |
|---|---|---|
| F | V | F |
| V | F | F |

Utilizando a idéia de tautologia, podemos clarear a **diferença** entre *"equivalente"* e *"logicamente equivalente"*:

Duas proposições p, q são logicamente equivalentes *se e somente se* $p \leftrightarrow q$ for uma tautologia,

ou seja,

$(p \Leftrightarrow q) \leftrightarrow (p \leftrightarrow q$ *é verdadeira*$)$.

Por exemplo, observe que $(p \rightarrow q) \leftrightarrow (\neg q \rightarrow \neg p)$ é uma equivalência lógica, isto é, $(p \rightarrow q) \Leftrightarrow (\neg q \rightarrow \neg p)$:

| p | q | ¬ p | ¬ q | (p → q) | (¬ q → ¬ p) | (p → q) ↔ (¬ q → ¬ p) |
|---|---|---|---|---|---|---|
| F | F | V | V | V | V | V |
| F | V | V | F | V | V | V |
| V | F | F | V | F | F | V |
| V | V | F | F | V | V | V |

Além do mais, utilizando a ideia de tautologia, podemos apresentar a definição de ***implicação lógica***:

$p \rightarrow q$ é uma implicação lógica *se* $p \rightarrow q$ for uma tautologia.

Neste caso, dizemos que *"p implica logicamente q" ou que "q é uma conseqüência lógica de p"*.

Uma implicação lógica deste tipo será denotada por:

$p \Rightarrow q$.

Além do mais, considerando este esquema de representação, note que:

$(p \Rightarrow q) \leftrightarrow (p \rightarrow q$ *é verdadeira*$)$.

Por exemplo, observe que $(p \wedge q) \rightarrow p$ é uma implicação lógica, ou seja, $(p \wedge q) \Rightarrow p$, e $p \rightarrow (p \wedge q)$ não é:

| p | q | (p ∧ q) | (p ∧ q) → p | p → (p ∧ q) |
|---|---|---|---|---|
| F | F | F | V | V |
| F | V | F | V | V |
| V | F | F | V | F |
| V | V | V | V | V |

Note que, se *"p implica logicamente q"*, e p é verdadeira, então q deve também ser verdadeira. Por exemplo, a posição 4 x 3 da tabela acima demonstra esta proposição (quando $p \wedge q$ é V então p é V). Observe que, no caso em que uma implicação não é uma implicação lógica esta proposição é violada, por exemplo, a posição 3 x 3 da tabela demonstra o que estamos mencionando (quando p é V então $p \wedge q$ é F).

Esta propriedade das tautologias, o fato delas serem sempre verdadeiras independentemente dos significados das subproposições envolvidas, é muito importante. Na realidade, as tautologias formam as regras pelas quais raciocinamos. A seguir apresentamos um conjunto destas regras. Em geral estas regras são utilizadas quando raciocinamos durante os processos de de-

monstração da validade de nossos argumentos (aguarde até a próxima seção para comprovar o que estamos mencionando):

Observe na lista de tautologias a seguir:

- **4-16** são equivalências lógicas enquanto **17-25** são implicações lógicas;
- **p**, **q**, **r** e **s** representam proposições;
- **t** representa tautologia;
- **c** representa contradição.

Assim, se quisermos raciocinar corretamente, deveremos empregar estas regras na obtenção de conseqüências lógicas de duas proposições conhecidas. Elas precisam estar incorporadas em nossa maneira de pensar. Observe atentamente todas as tautologias e tente compreender porque elas são sempre verdadeiras, independentemente do que p, q, r e s estejam representando.

Por exemplo, se uma pessoa diz *"esta blusa é feita de algodão ou de seda"* e, em seguida, uma outra pessoa observa melhor e diz *"ela não é feita de algodão"* o que você poderia dizer a respeito? Raciocinando corretamente, por exemplo, representando *"esta blusa é feita de algodão"* por p e "esta blusa é feita de seda" por q e empregando a tautologia 22, você poderia dizer que *"a blusa é feita de seda"* pois *"se a blusa é feita de algodão ou de seda, e ela não é feita de algodão então a blusa é feita de seda"*, ou seja, $((p \vee q) \wedge \neg p) \Rightarrow q$.

Além do **silogismo disjuntivo** descrito acima, vale apresentar um tipo de raciocínio muito comum em nossas conversações ordinárias, ou seja, o raciocínio ***modus ponens***. Por exemplo, se é verdade que uma pessoa disse, em algum momento, "se eu fizer os exercícios de fixação, é porque eu gostei da aula" e, atualmente, essa mesma pessoa diz "eu vou fazer os exercícios de fixação", o que poderíamos concluir a respeito de nossa amiga? Considerando a tautologia 19, $(p \wedge (p \rightarrow q)) \Rightarrow q$, poderíamos afirmar que "Danielle gostou da aula".

| | | |
|---|---|---|
| **1.** | $p \vee \neg p$ | |
| **2.** | $\neg ( p \wedge \neg p)$ | |
| **3.** | $p \rightarrow p$ | |
| **4.** | a) $p \leftrightarrow (p \vee p)$ | "idempotent laws" |
| | b) $p \leftrightarrow (p \wedge p)$ | |
| **5.** | $\neg \neg p \leftrightarrow p$ | "double negation" |

| | | |
|---|---|---|
| **6.** | a) $(p \vee q) \leftrightarrow (q \vee p)$ | "commutative laws" |
| | b) $(p \wedge q) \rightarrow (q \wedge p)$ | |
| | c) $(p \leftrightarrow q) \leftrightarrow (q \leftrightarrow p)$ | |
| **7.** | a) $(p \vee (q \vee r) \leftrightarrow ((p \vee q) \vee r)$ | "associative laws" |
| | b) $(p \wedge (q \wedge r) \leftrightarrow ((p \wedge q) \wedge r)$ | |
| **8.** | a) $(p \wedge (q \vee r) \leftrightarrow ((p \wedge q) \vee (p \wedge r))$ | "distributive laws" |
| | b) $(p \vee (q \wedge r) \leftrightarrow ((p \vee q) \wedge (p \vee r))$ | |
| **9.** | a) $(p \vee c) \leftrightarrow p$ | "identity laws" |
| | b) $(p \wedge c) \leftrightarrow c$ | |
| | d) $(p \vee t) \leftrightarrow t$ | |
| | d) $(p \wedge t) \leftrightarrow p$ | |
| **10.** | a) $\neg (p \wedge q) \leftrightarrow (\neg p \vee \neg q)$ | "DeMorgan's laws" |
| | b) $\neg (p \vee q) \leftrightarrow (\neg p \wedge \neg q)$ | |
| **11.** | a) $(p \leftrightarrow q) \leftrightarrow ((p \rightarrow q) \wedge (q \rightarrow p))$ | "equivalence" |
| | b) $(p \leftrightarrow q) \leftrightarrow ((p \wedge q) \vee (\neg p \wedge \neg q))$ | |
| | c) $(p \leftrightarrow q) \leftrightarrow (\neg p \leftrightarrow \neg q)$ | |
| **12.** | a) $(p \rightarrow q) \leftrightarrow (\neg p \vee q)$ | "implication" |
| | b) $\neg (p \rightarrow q) \leftrightarrow (p \wedge \neg q)$ | |
| **13.** | $(p \rightarrow q) \leftrightarrow (\neg q \rightarrow \neg p)$ | "contrapositive" |
| **14.** | $(p \rightarrow q) \leftrightarrow ((p \wedge \neg q) \rightarrow c)$ | "reductio ad absurdum" |
| **15.** | a) $(p \rightarrow r) \wedge (q \rightarrow r) \leftrightarrow ((p \vee q) \rightarrow r)$ | |
| | b) $((p \rightarrow q) \wedge (p \rightarrow r)) \rightarrow (p \rightarrow (q \wedge r))$ | |
| **16.** | $((p \wedge q) \rightarrow r) \rightarrow (p \rightarrow (q \rightarrow r))$ | "exportation law" |
| **17.** | $p \rightarrow (p \vee q)$ | "addition" |
| **18.** | $(p \wedge q) \rightarrow p$ | "simplification" |
| **19.** | $(p \wedge (p \rightarrow q)) \rightarrow q$ | "modus ponens" |
| **20.** | $((p \rightarrow q) \wedge \neg q) \rightarrow \neg p$ | "modus tollens" |
| **21.** | $((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$ | "hypothetical syllogism" |
| **22.** | $((p \vee q) \wedge \neg p) \rightarrow q$ | "disjunctive syllogism" |
| **23.** | $(p \rightarrow c) \rightarrow \neg p$ | "absurdity" |
| **24.** | $((p \rightarrow q) \wedge (r \rightarrow s)) \rightarrow ((p \vee r) \rightarrow (q \vee s))$ | |
| **25.** | $(p \rightarrow q) \rightarrow ((p \vee r) \rightarrow (q \vee r))$ | |

Com relação à lista de tautologias, é importante que tentemos assimilar as formas das tautologias tal que possamos reconhecer quando às estivermos utilizando. Além do mais, é importante reconhecermos o raciocínio incorreto; isto é, quando estivermos considerando incorretamente uma nova proposição como sendo conseqüência lógica de duas proposições conhecidas.

## Para refletir

**1.** Determine quais das seguintes formas abaixo têm a forma de uma das tautologias apresentadas na lista em anexo (por exemplo, $(\neg q \wedge p) \rightarrow \neg q$ tem a forma de 18).

a) $\neg q \rightarrow (\neg q \vee \neg p)$.

b) $q \rightarrow (q \wedge \neg p)$.

c) $(r \rightarrow \neg p) \leftrightarrow (\neg r \vee \neg p)$.

d) $(p \rightarrow \neg q) \leftrightarrow \neg (\neg p \rightarrow q)$.

e) $(\neg r \rightarrow q) \leftrightarrow (\neg q \rightarrow r)$.

f) $(p \rightarrow (\neg r \vee q)) \leftrightarrow ((r \wedge \neg q) \rightarrow \neg p)$.

g) $r \rightarrow \neg (q \wedge \neg r)$.

h) $((\neg q \vee p) \wedge q) \rightarrow p$.

**2.** Dê exemplos de proposições ou fale porque o exemplo não existe:

a) Uma implicação lógica com uma conclusão falsa.

b) Uma implicação lógica com uma conclusão verdadeira.

c) Uma implicação lógica com uma hipótese verdadeira e uma conclusão falsa.

**3.** Indique quais das seguintes proposições são verdadeiras:

a) $(p \rightarrow (q \vee r)) \Rightarrow (p \rightarrow q)$.

b) $((p \vee q) \rightarrow r) \Rightarrow (p \rightarrow r)$.

c) $(p \vee (p \wedge q)) \Leftrightarrow p$.

d) $((p \rightarrow q) \wedge \neg p) \Rightarrow \neg q$.

**4.** Quais das seguintes formas são tautologias, contradições ou nem uma e nem outra coisa?

a) $(p \wedge \neg q) \rightarrow (q \vee \neg p)$.

b) $\neg p \rightarrow p$.

c) $\neg p \leftrightarrow p$.

d) $(p \wedge \neg p) \rightarrow p$.

e) $(p \wedge \neg p) \rightarrow q$.

f) $(p \wedge \neg q) \leftrightarrow (p \rightarrow q)$.

g) $[(p \rightarrow q) \leftrightarrow r] \leftrightarrow [p \rightarrow (q \leftrightarrow r)]$.

**5.** Quais das seguintes formas abaixo são corretas?

a) $(p \leftrightarrow q) \Rightarrow (p \rightarrow q)$.

b) $(p \rightarrow q) \Rightarrow (p \leftrightarrow q)$.

c) $(p \rightarrow q) \Rightarrow q$.

**6.** Será que $\rightarrow$ é associativa; isto é, será que $((p \rightarrow q) \rightarrow r) \Leftrightarrow (p \rightarrow (q \rightarrow r))$?

**7.** Será que $\leftrightarrow$ é associativa; isto é, será que $((p \leftrightarrow q) \leftrightarrow r) \Leftrightarrow (p \leftrightarrow (q \leftrightarrow r))$?

**8.** Quais das seguintes proposições verdadeiras são tautologias?

a) Se 2 + 2 = 4 então 5 é impar.

b) 3 + 1 = 4 e 5 + 3 = 8 implica 3 + 1 = 4.

c) 3 + 1 = 4 e 5 + 3 = 8 implica 3 + 2 = 5.

d) Vermelho é amarelo ou vermelho não é amarelo.

e) Vermelho é amarelo ou vermelho é vermelho.

f) 4 é impar ou 2 é par e 2 é impar implica que 4 é impar.

g) 4 é impar ou 2 é par e 2 é impar implica que 4 é par.

**9.** Quais das seguintes conclusões são conseqüências lógicas do conjunto de proposições $p \vee q, r \rightarrow \neg q, \neg p$?

a) q. d) $\neg$ r.

b) r. e) $\neg (\neg q \wedge r)$.

c) $\neg p \vee s$. f) $q \rightarrow r$.

## 5. Argumentos

**Argumentos** são proposições descritas no formato de implicações. Em geral, o antecedente de um argumento é uma conjunção de subproposições, simples ou compostas, e o conseqüente é uma subproposição, do mesmo tipo do antecedente. Sendo assim, podemos representar os argumentos de uma pessoa através da seguinte **forma geral:**

$$(p_1 \wedge p_2 \wedge \ldots \wedge p_n) \rightarrow q;$$

Onde $\mathbf{p_1}$, $\mathbf{p_2}$, ..., $\mathbf{p_n}$ **e q** representam proposições em linguagem natural, que estão ligadas por meio de conectivos.

Por exemplo, o quadro abaixo apresenta um argumento em Português e sua representação em Linguagem Proposicional. Observe que o antecedente deste argumento é uma conjunção de três proposições compostas e o conseqüente é uma proposição simples:

| Argumento em Português |
|---|
| $p_1$ - "Se Platão estiver disposto a visitar Sórates então Sócrates está disposto a visitar Platão" |
| $p_2$ - "Se Sócrates está disposto a visitar Platão então Platão não está disposto a visitar Sócrates" |
| $p_3$ - "Se Sócrates não está disposto a visitar Platão então Platão está disposto a visitar Sócrates" |
| q - "Sócrates está disposto a visitar Platão" |

| Argumento em Linguagem Proposicional |
|---|
| Considerando: |
| p - "Sócrates está disposto a visitar Platão" e |
| r - "Platão está disposto a visitar Sócrates", |
| temos: |
| $p_1$ - $r \rightarrow p$ |
| $p_2$ - $p \rightarrow \neg r$ |
| $p_3$ - $\neg p \rightarrow r$ |
| q - p. |
| Na forma geral: $((r \rightarrow p) \wedge (p \rightarrow \neg r) \wedge (\neg p \rightarrow r)) \rightarrow p$ |

Vale ressaltar, os **teoremas matemáticos** podem ser representados no formato dos argumentos. Além do mais, observe a definição de **Programa em Lógica** e **Consulta a um Programa**. Um **Programa em Lógica** é um conjunto de proposições (programa = teoria) a respeito de um mundo particular, e uma **Consulta a um Program**a em lógica é uma proposição (consulta = conseqüência lógica, ou não, da teoria) a respeito deste mundo.

Isto nos sugere uma analogia entre argumentos, e programas e consultas. Na realidade, esta analogia e o entendimento do processo de demonstração da validade de argumentos são úteis para a assimilação da idéia de se programar em Lógica. De acordo com esta analogia, um programa em Lógica pode ser representado pelo conjunto de fbfs que compõem o antecedente da forma geral de um argumento correspondente, ou seja:

$$p_1, p_2, ..., p_n;$$

e a consulta ao programa pela fbf que compõe o conseqüente do argumento, ou seja: ***q.***

Por exemplo, considerando o argumento apresentado no início desta seção, temos o seguinte programa e consulta em Lógica Proposicional:

| Programa em Linguagem Lógica Proposicional |
|---|
| $p_1$ - $r \rightarrow p$ |
| $p_2$ - $p \rightarrow \neg r$ |
| $p_3$ - $\neg p \rightarrow r$ |
| **Consulta em Linguagem Lógica Proposicional** |
| q - p ? |

Na realidade, a ideia de Programação em Lógica fundamenta-se na analogia entre os argumentos, e os programas e as consultas. Nesta disciplina, exploramos esta analogia programando em Linguagem Formal Lógica. Mas, antes disso, ainda precisamos considerar o processo de demonstração de validade de argumentos e os meios que podem ser empregados para esta finalidade.

## 5.1 Validade de Argumentos

O que precisamos fazer para ganhar uma argumentação? Ou seja, o que precisamos fazer para demonstrar que um certo argumento é válido? Primeiro, devemos agir de uma maneira convincente, ou seja, devemos convencer a(s) pessoa(s) a respeito da verdade lógica de nossa posição. Por exemplo, você pode começar perguntando para uma pessoa: você aceita que $p_1, p_2, ..., p_n$ são verdadeiras? Se a resposta for sim, você pode dizer: então segue que $q$, também, é verdadeira. Em seguida, para você ganhar a argumentação você deve provar que $(p_1 \wedge p_2 \wedge ... \wedge p_n) \rightarrow q$ *é uma tautologia, ou seja, demonstrando que:*

$$(p_1 \wedge p_2 \wedge ... \wedge p_n) \Rightarrow q.$$

Mais precisamente, devemos mostrar que se a conjunção $p_1 \wedge p_2 \wedge ... \wedge p_n$ for verdadeira (V), então q é também verdadeira (V); neste caso, $(p_1 \wedge p_2 \wedge ... \wedge p_n) \rightarrow q$ é uma tautologia (na tabela verdade do argumento não existe uma situação em que a conjunção das hipóteses é V e a conclusão é F) e q é conseqüência lógica de $p_1 \wedge p_2 \wedge ... \wedge p_n$.

Na **Lógica Proposicional**, podemos empregar, por exemplo, os seguintes meios para demonstrar que um argumento é válido:

- Tabelas Verdade,
- Princípio da Demonstração e
- Extensão do Princípio da Demonstração.

Além do mais, todos os meios empregáveis na demonstração da validade de argumentos, podem ser empregados na obtenção de respostas para consultas. Na realidade, nas próximas quatro subseções utilizaremos três meios, utilizados na demonstração da validade de argumentos, para implementar a idéia de Programação em Lógica Proposicional. Assim, dividimos esta Seção em mais três subseções:

**a)** Validade e as Tabelas Verdade

**b)** Validade e o Princípio da Demonstração

**c)** Validade e a Extensão do Princípio da Demonstração

### a) Validade e as Tabelas Verdade

Utilizando uma tabela verdade podemos provar a validade de um argumento e, conseqüentemente, programar em Lógica Proposicional. Por exemplo, considere o argumento a respeito do mundo de Sócrates e Platão e o método da Tabela Verdade respondendo *'sim'* para a consulta *"Sócrates está disposto a visitar Platão?"*:

| **Hipóteses do Argumento** |
|---|
| $r \rightarrow p$<br>$p \rightarrow \neg r$<br>$\neg p \rightarrow r$ |
| **Conclusão do Argumento** |
| p |
| **Método de Demonstração da Validade do Argumento** |

Tabela Verdade

| p | r | ¬ p | ¬ r | $r \rightarrow p$ | $p \rightarrow \neg r$ | $\neg p \rightarrow r$ |
|---|---|---|---|---|---|---|
| F | F | V | V | V | V | V |
| F | V | V | F | F | V | V |
| V | F | F | V | V | V | V |
| V | V | F | F | V | F | V |

| $(r \rightarrow p) \wedge (p \rightarrow \neg r) \wedge (\neg p \rightarrow r)$ | $(r \rightarrow p) \wedge (p \rightarrow \neg r) \wedge (\neg p \rightarrow r) \rightarrow p$ |
|---|---|
| F | V |
| F | V |
| V | V |
| F | V |

Como sempre que $(r \rightarrow p)\ (p \rightarrow \neg r) \wedge (\neg p \rightarrow r)$ é V, p é V (ver terceira linha da tabela verdade), então $(r \rightarrow p) \wedge (p \rightarrow \neg r) \wedge (\neg p \rightarrow r) \rightarrow p$ é uma tautologia, conseqüentemente, um argumento válido, ou seja, $(r \rightarrow p) \wedge (p \rightarrow \neg r) \wedge (\neg p \rightarrow r) \Rightarrow p$.

Logo, a **Resposta** é '**sim**', ou seja, "Sócrates está disposto a visitar Platão".

Além do mais, observe abaixo o Método da Tabela Verdade demonstrando que um argumento não é válido, ou seja, respondendo que 'a consulta não é conseqüência lógica do programa':

| Hipóteses do Argumento |
|---|
| "Se 2 + 2 = 4 então 3 + 5 = 7"<br>"2 + 2 ≠ 4" |
| **Conclusão Argumento** |
| "3 + 5 ≠ 7 ?" |

| Hipóteses do Argumento |
|---|
| Considerando:<br>p - "2 + 2 ≠ 4" e<br>q - "3 + 5 = 7",<br>temos:<br>$p_1$ - $\neg p \rightarrow q$<br>$p_2$ - p |
| **Conclusão do Argumento** |
| q - ¬q.<br>Assim, argumento na forma geral: $((\neg p \rightarrow q) \wedge p) \rightarrow \neg q$ |

| Método de Demonstração da Validade do Argumento | | | | | | |
|---|---|---|---|---|---|---|
| **p** | **q** | **¬ p** | **¬ q** | **¬p → q** | **(¬p → q) ∧ p** | **((¬p → q) ∧ p) → ¬q** |
| F | F | V | V | F | F | V |
| F | V | V | F | V | F | V |
| V | F | F | V | V | V | V |
| V | V | F | F | V | V | F |

Como nem sempre que $(\neg p \rightarrow q) \wedge p$ é V, ¬q é V (ver quarta linha da tabela verdade), então $((\neg p \rightarrow q) \wedge p) \rightarrow \neg q$ não é uma tautologia, conseqüentemente, um argumento não-válido, ou seja, ¬q não é conseqüência lógica de $(\neg p \rightarrow q) \wedge p$.

Neste caso, consideraremos que a Resposta é '3 + 5 ≠ 7 não é conseqüência lógica do programa'.

Observe que se o número de subproposições diferentes envolvidas em um argumento/programa-consulta for muito grande então pode ser inconveniente checar a validade de argumentos/responder a uma consulta utilizando uma tabela verdade, pois esta tabela pode conter muitas linhas. Esta restrição impossibilita a utilização generalizada de Tabelas Verdade na demonstração da validade de argumentos e/ou na Programação em Lógica Proposicional.

## Para refletir

**1.** Onde for possível dê exemplos; se não for possível diga porque:

a) Um argumento não-válido com uma conclusão falsa.

b) Um argumento válido com uma conclusão verdadeira.

c) Um argumento não-válido com uma conclusão verdadeira.

d) Um argumento válido com uma conclusão falsa.

e) Um argumento válido com uma hipótese verdadeira e uma conclusão falsa.

f) Um argumento não-válido com uma hipótese verdadeira e uma conclusão falsa.

g) Um argumento válido com uma hipótese falsa e uma conclusão verdadeira.

**2.** Determine a validade dos argumentos descritos em Linguagem Lógica Proposicional, usando tabelas verdade:

a) $p \rightarrow q$
$\neg p \vee q$
$q \rightarrow p$

b) $p \vee q$
$r \rightarrow q$
$q$
$\neg r$

c) $p \vee \neg q$
$\neg p$
$\neg q$

d) $q \vee \neg p$
$\neg q$
$p$

e) $\neg p$
$p \rightarrow q$

f) $(p \wedge q)$ □ $(r \wedge s)$
$\neg r$
$\neg p \vee \neg q$

g) $p \rightarrow q$
$\neg q \rightarrow \neg r$
$s \rightarrow (p \vee r)$
$s$
$q$

h) $p \vee q$
$q \rightarrow \neg r$
$\neg r \rightarrow \neg p$
$\neg(p \wedge q)$

i) $p \rightarrow q$
$\neg r$ $\neg q$
$r \rightarrow \neg p$
$\neg p$

j) $p \rightarrow \neg p$
$\neg p$

k) $p \vee q$
$p \rightarrow r$
$\neg r$
$q$

l) $p$
$q \rightarrow \neg p$
$\neg q \rightarrow (r \vee \neg s)$
$\neg r$
$\neg s$

**3.** A seguir, apresentamos alguns argumentos. Determine a validade dos argumentos/responda às consultas descritas em Português, utilizando tabelas verdade:

a) Hipóteses do Argumento 1

"Se o dia está bonito então vou à praia"

"Não vou à praia"

Conclusão do Argumento 1

"O dia não está bonito ?"

b) Hipóteses do Argumento 2

"Se Nazaré Coelho está na Universidade então Pedro está no hospital e José mudou de emprego"

"José mudou de emprego"

"Pedro está no hospital"

Conclusão do Argumento 2

"Nazaré Coelho está na Universidade ?"

c) Hipóteses do Argumento 3

"Se João descobre a conspiração e der valor a sua vida, então abandonará o país"

"João dá valor a sua vida"

Conclusão do Argumento 3

"Se João descobre a conspiração então abandonará o país ?"

d) Hipóteses do Argumento 4

"Se Talita conseguir arranjar um carro emprestado e for pela auto-estrada, então chegará antes de terminar o prazo"

"Talita chegará antes de terminar o prazo"

Conclusão do Argumento 4

"Se Talita consegue arranjar um carro emprestado então vai pela auto-estrada ?"

e) Hipóteses do Argumento 5

"Se Gerson não está em condições então Bira será zagueiro de área ou Miguel será o zagueiro de área"

"Bira não é o zagueiro de área"

Conclusão do Argumento 5

"Se Miguel não é o zagueiro de área então Gerson está em condições ?"

f) Hipóteses do Argumento 6

"Se Almir apoia o presidente então Jader apoia o novo candidato"

"Se Jader apoia o novo candidato então Hélio abandonará o partido"

"Se Hélio abandona o partido então Almir não apoia o presidente"

Conclusão do Argumento 6

"Almir não apoia o presidente ?"

g) Hipóteses do Argumento 7

"Se Paulo se retira da reunião então Basílio será nomeado ou Clara ficará desapontada"

"Basílio não será nomeado"

Conclusão do Argumento 7

"Se Paulo se retira da reunião então Clara ficará desapontada ?"

## 6. Validade, Programação e o Princípio da Demonstração

Além da tabela verdade, também, podemos provar a validade de um argumento empregando o Princípio da Demonstração. Veja o enunciado deste Princípio:

"Uma demonstração que o argumento $(p_1 \wedge p_2 \wedge \ldots \wedge p_n) \rightarrow q$ é válido, é uma seqüência de proposições $s_1, s_2, \ldots, s_k$ tal que $s_k$ (última proposição na seqüência) = $q$ (a conclusão) e cada $s_i$, $1 \leq i \leq k$ satisfaz um ou mais dos requisitos:

**a)** $s_i$ é uma das hipóteses do argumento ($p_1, p_2, \ldots, p_n$,)

**b)** $s_i$ é uma tautologia

**c)** $s_i$ é uma conseqüência lógica de proposições recentes na seqüência."

Por exemplo, observe o Princípio da Demonstração mostrando que o argumento sobre o mundo de Sócrates e Platão é válido, ou seja, que a resposta à consulta é *"sim"*:

| **Hipóteses do Argumento** |
|---|
| $p_1$ - $r \rightarrow p$<br>$p_2$ - $p \rightarrow \neg r$<br>$p_3$ - $\neg p \rightarrow r$ |
| **Conclusão do Argumento** |
| q - p. |
| **Método de Demonstração da Validade do Argumento** |
| Princípio da Demonstração<br>$s_1$ = $r \rightarrow p$ $s_2$ = $p \rightarrow \neg r$<br>$s_3$ = $(r \rightarrow p) \wedge (p \rightarrow \neg r)$ $s_4$ = Tautologia 21 - $((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r))$<br>$s_5$ = $(r \rightarrow \neg r)$ $s_6$ = Tautologia 12.a) - $(p \rightarrow q) \leftrightarrow (\neg p \vee q)$<br>$s_7$ = $\neg r \vee \neg r$ $s_8$ = $(\neg r \vee \neg r) \leftrightarrow \neg r$<br>$s_9$ = $\neg r$ $s_{10}$ = $\neg p \rightarrow r$<br>utilizando Tautologia 20 –<br>$((p \rightarrow q) \wedge \neg q) \rightarrow \neg p$<br>$s_{11}$ = $\neg(p)$ $s_{12}$ = Tautologia 5 - $\neg\neg p \leftrightarrow p$<br>$s_{13}$ = p |

Assim, como a última proposição na seqüência, $s_{13}$, é a conclusão do argumento/consulta ao programa então o argumento é válido e, consequentemente, a resposta é ***'sim'***.

## Para refletir

1. Para cada um dos argumentos que você considerou como válidos na Atividade de Avaliação 3 na Seção "Validade e as Tabelas Verdade", utilize a Adaptação do Princípio da Demonstração para a Programação em Lógica e mostre que cada uma das respostas correspondentes é ***'sim'***.

## 7. Validade, Programação e a Extensão do Princípio

A Extensão do Princípio da Demonstração é uma outra forma de se provar a validade de argumentos e, conseqüentemente, de se programar em lógica. É um método de prova indireta ou **prova por contradição**. A Tautologia 14 - $(p \rightarrow q) \leftrightarrow ((p \wedge \neg q) \rightarrow \mathbf{c})$, fundamenta a prova por contradição. Podemos perceber melhor este método empregando esta tautologia à forma geral do argumento, ou seja:

$$((p_1 \wedge p_2 \wedge \ldots \wedge p_n) \rightarrow q) \leftrightarrow ((p_1 \wedge p_2 \wedge \ldots \wedge p_n \wedge \neg q) \rightarrow \mathbf{c}).$$

Assim, de acordo com a equivalência lógica acima, a extensão nos diz que para mostrarmos que o argumento é válido/que a resposta é ***'sim'***, devemos mostra que a conjunção das hipóteses do argumento/fbfs do programa com a negação da conclusão do argumento/consulta ao programa implica em uma contradição, ou seja:

$$(p_1 \wedge p_2 \wedge \ldots \wedge p_n \wedge \neg q) \rightarrow \mathbf{c}.$$

Mais especificamente, aplicando a Extensão do Princípio da Demonstração para provar a validade de um argumento/responder à consulta, o processo de prova deve terminar quando a última proposição na seqüência, $s_k$, for uma contradição e, somente neste caso, a resposta será ***'sim'***. Observe a Extensão mostrando que o argumento sobre o mundo de Sócrates e Platão é válido, ou seja, que a resposta à consulta é *"sim"*:

| **Hipóteses do Argumento** |
|---|
| $p_1$ - $p \vee q$ |
| $p_2$ - $q \rightarrow \neg p$ |
| $p_3$ - $p \rightarrow q$ |
| **Conclusão do Argumento** |
| q - q. |
| **Negação da Conclusão do Argumento** |
| ¬q - ¬q. |
| **Novas Hipóteses do Argumento** |
| $p_1$ - $p \vee q$ |
| $p_2$ - $q \rightarrow \neg p$ |
| $p_3$ - $p \rightarrow q$ |
| $p_4$ - ¬q |
| **Nova Conclusão do Argumento** |
| q - contradição. |

**Método de Demonstração da Validade do Argumento**

Extensão do Princípio da Demonstração

$s_1 = \neg q$ $s_2 = p \vee q$

utilizando Tautologia 22 –

$(p \vee q) \wedge \neg p) \rightarrow q$

$s_3 = p$ $s_4 = p \rightarrow q$

utilizando Tautologia 19 –

$(p \wedge (p \rightarrow q)) \rightarrow q$

$s_5 = q$ $s_6 = \neg q$

$s_7 = q \wedge \neg q$ $s_8 = (q \wedge \neg q) \leftrightarrow$ contradição

$s_9 =$ contradição

Observe, na seqüência de proposições ( $s_1$ , $s_2$ , ... , $s_9$ ), cada uma das proposições satisfaz pelo menos um dos requisitos descritos no enunciado do Princípio da Demonstração, ou seja:

**a)** $s_1$ , $s_2$ , $s_4$ , $s_6$ - são hipóteses do argumento/fbf's do programa, inclusive a consulta negada, $s_6$;

**b)** $s_8$ - é uma tautologia;

**c)** $s_3$ , $s_5$ , $s_7$, $s_9$ - são conseqüências lógicas de proposições recentes na seqüência.

Assim, como a última proposição na seqüência, $s_9$, é a nova conclusão do argumento/nova consulta ao programa, isto é, a negação da conclusão do argumento/negação da consulta ao programa é uma **contradição**, então o argumento é válido, consequentemente a resposta é ***'sim'***.

Diferentemente da Tabela Verdade, o Princípio da Demonstração e a Extensão do Princípio não demonstram que um argumento é não-válido; o fato de não se demonstrar a validade não garante que o argumento seja não-válido.

Para se demonstrar a não validade de um argumento, além da Tabela Verdade, podemos tentar encontrar um contra-exemplo, ou seja, uma interpretação para as subproposições envolvidas no argumento tal que suas hipóteses, $p_1$ ,$p_2$ , ... , $p_n$, sejam todas verdadeiras e a conclusão, $q$, seja falsa. Neste caso, o argumento é não-válido pois, $(V \wedge V \wedge ... \wedge V) \rightarrow F$ é $F$. Por exemplo, o argumento

$p_1$ - $p \rightarrow q$

$p_2$ - $\neg p \vee q$

$q$ - $q \rightarrow p$ ,

Mais precisamente, $( (p \ \ q) \wedge (\neg p \vee q) ) \rightarrow (q \rightarrow p)$, é não-válido pois se, por exemplo, $p$ for interpretado como *"2 < 1"* (uma subproposição ***F***), e $q$ como *" 3 > 2"* (uma suproposição ***V***), todas as hipóteses do argumento serão verdadeiras,

$p_1$ - $F \rightarrow V$ é $V$

$p_2$ - $V \vee V$ é $V$,

e a conclusão do argumento é falsa,

$q$ - $V \rightarrow F$ é $F$;

Logo, estas interpretações produzem um argumento não-válido, $(V \wedge V) \rightarrow F$ é $F$.

Assim,

*"se 2 < 1 então 3 > 2"*

*"2 > 1 ou 3 >2"*

*"se 3 >2 então 2 < 1"*

É um **contra-exemplo** para o argumento.

## Atividades de avaliação

1. Para cada um dos argumentos que você considerou como válidos na Atividade de Avaliação 3 na Seção "Validade e as Tabelas Verdade", utilize a Extensão do Princípio da Demonstração e mostre que cada uma das respostas correspondentes é ***'sim'***.

2. Para cada um dos argumentos que você considerou como não-válidos na Atividade de Avaliação 3 na Seção "Validade e as Tabelas Verdade", encontre um contra-exemplo.

Parte 2

# Lógica de Predicados

Capítulo 3

# Linguagem lógica de predicados

## Objetivos

- Conhecer a linguagem formal Lógica de Predicados, assim como gerar fórmulas bem formadas nessa linguagem, a atribuição de valor verdade às fórmulas envolvendo quantificadores, algumas implicações e equivalências lógicas que podem ser empregadas durante o processo de raciocínio.
- Aplicar o conhecimento adquirido nos processos de representação de programas em Lógica de Predicados e de raciocínio automático na obtenção de respostas para consultas realizadas aos programas.

## Introdução

Uma **Teoria** consiste de um conjunto de proposições acerca de um universo de discurso ***(U)***. Considere o exemplo abaixo:

**Catálogo:**

$\mathbf{p_1}$. Soma.for é um programa FORTRAN

$\mathbf{p_2}$. Soma.pas é um programa PASCAL

$\mathbf{p_3}$. Soma.pro é um programa PROLOG

$\mathbf{p_4}$. Todo programa FORTRAN é um programa procedimental

$\mathbf{p_5}$. Todo programa PASCAL é um programa procedimental

$\mathbf{p_6}$. Todo programa PROLOG é um programa declarativo

***U*** *= {Soma.for. Soma.pas, Soma.pro, FORTRAN, PASCAL, PROLOG}*

Um **Sistema Formal** consiste de uma **linguagem formal** e de uma **abstração adequada para os princípios** usados para decidir quando uma proposição é conseqüência lógica de outras.

A **Lógica Proposicional** é um exemplo de sistema formal.

Nesse contexto, a **Lógica de Predicados de Primeira Ordem** é uma extensão da Lógica Proposicional que pode ser caracterizada como um **sistema formal** apropriado **à definição de teorias** acerca de diversos universos de discurso.

A **Linguagem Lógica de Predicados** permite a representação de proposições que não podem ser representadas razoavelmente na Linguagem Lógica Proposicional.

**Exemplo:**

| Proposição | Lógica Proposicional | Lógica de Predicados |
|---|---|---|
| $p_1$. Sócrates é um homem | p | homem(Sócrates) |
| $p_2$. Platão é um homem | q | homem(Platão) |
| $p_3$. Todos os homens são mortais | r | $\forall x$:(homem(x) $\rightarrow$ mortal(x)) |

A **Resolução para Lógica de Predicados** permite que se decida se uma proposição é conseqüência lógica de outras.

## 1. Linguagem Lógica de Predicados

O **Alfabeto** ***(A)*** da Linguagem Lógica de Predicados consiste de:

**a)** uma série de **símbolos lógicos**:

- variáveis,
- conectivos lógicos,
- quantificadores,
- pontuação e
- símbolo de igualdade;

**b)** uma série de **símbolos não-lógicos**:

- constantes,
- símbolos predicativos e
- símbolos funcionais.

**constantes** - servem para denotar indivíduos específicos do universo de discurso

| Símbolo Lógicos | Símbolos Não-lógicos |
|---|---|
| pontuação: ( , )<br>símbolo de igualdade (opcional): =<br>variáveis: x, y, ...<br>conectivos lógicos: ¬, ∧, ∨, □, ↔<br>quantificadores: ∃, ∀ | constantes: FORTRAN, PASCAL, PROLOG, ...<br>símbolos predicativos: programa, procedimental, declarativo, ...<br>símbolo funcionais: sen, cos, ... |

**símbolos predicativos** - servem para denotar relações entre indivíduos.

**símbolos predicativos** - servem para denotar funções sobre indivíduos.

**quantificador universal** - "para todo .."
**quantificador existencial** - ëxiste ..."

**variáveis** - servem para denotar indivíduos arbitrários do universo de discurso.

Um **Termo** particular da linguagem pode ser:

- uma variável ou
- uma constante ou
- uma expressão da forma $f(t_1, t_2, t_m)$, onde

$f$ - é um símbolo funcional admitindo $m$ argumentos e

$t_1, t_2, ..., t_m$ - são termos.

As **Regras Sintáticas** da linguagem definem as sentenças da linguagem como sendo **fórmulas** de dois tipos:

a) fórmula atômica da forma $p(t_1, t_2, t_n)$, onde

$p$ - é um símbolo predicativo admitindo $n$ argumentos e

$t_1, t_2, ..., t_n$ - são termos;

b) expressão obtida compondo-se fórmulas

- através dos conectivos lógicos ou
- prefixando-se fórmulas com quantificadores.

Considere o exemplo a seguir:

**Termos -** permitem denotar indivíduos do domínio de discurso diretamente através de seus nomes ou indiretamente através de funções.

| Termos | Fórmulas |
|---|---|
| variável: x, y, ... | fórmula atômica: programa(Soma.for, FORTRAN), procedimental(x), ... |
| constantes: 0, 1, Soma.for ... | fórmula: programa(Soma.for, FORTRAN) $\wedge$ programa(Soma.pas, PASCAL) |
| expr. func.: sen(x), cos(y), ... | fórmula: $\forall$x:(programa(x, FORTRAN) □ procedimental(x)), ... |

**Fórmulas** (não-atômicas) - permitem expressar propriedades das relações entre indíviduos do universo de discurso.

**Fórmulas atômicas -** permitem expressar relações entre indivíduos do universo de discurso e as propriedades destes.

A **Linguagem Lógica de Predicados** é o **conjunto de fórmulas bem formadas** sobre ***A***.

As **Regras Semânticas** da linguagem capturam o **significado das fórmulas** definidas sobre o alfabeto associando um dos valores verdade *V* ou *F*.

Veja o exemplo abaixo:

| Universo | Alfabeto | Significado |
|---|---|---|
| U = conj. reais | A = {maior, x, y, ...} $\cup$ U | maior(x,y) = V sss (x, y) $\in$ {(m,n) $\in$ UxU $\mid$ m > n} |
| | | maior(2,1) = V |

# Capítulo 4

# Quantificadores

## Introdução

Uma **Proposição** é uma sentença declarativa que é verdadeira ou falsa.

**Exemplo:**

| Proposição | Valor Verdade |
|---|---|
| "2 < 3" | Verdadeira |

Uma **Função Proposicional** é uma sentença declarativa contendo uma ou mais variáveis que se torna uma proposição quando valores particulares (interpretações) são atribuídos às variáveis.

O **Domínio de Discurso** de uma função proposicional é o conjunto de valores possíveis para as variáveis da função. Supondo-se *D* o domínio de discurso de uma função proposicional *p*, podemos tornar *p* uma proposição **substituindo** vários membros de *D* em *p*

Considere o seguinte exemplo:

| F. Prop. | $x \in D1 = \{1,2,3\}$ | $y \in D2 = \{2,4\}$ | Prop. | Representação | V. Verdade |
|---|---|---|---|---|---|
| "x < 3" | 1 | - | "1 < 3" | p(1) | verdadeira |
| "x < 3" | 3 | - | "2 < 3" | p(3) | falsa |
| "x < y" | 2 | 4 | "2 < 4" | q(2, 4) | verdadeira |
| "x < y" | 3 | 2 | "3 < 2" | q(3, 2) | falsa |

## 1. Quantificando a função proposicional p

**Maneiras de se quantificar** uma função proposicional:

- prefaciar a função proposicional com *"**para todo x em D**"* e
- prefaciar a função proposicional com *"**existe um x em D tal que**"*

Veja o exemplo:

| F. Prop. | Representação | Proposição | Representação |
|---|---|---|---|
| "x < 3" | p(x) | "para todo x em $D_1$, p(x)" | ∃x:p(x) |
| "x < 3" | p(x) | "existe x em $D_1$ tal que p(x)" | ∀x:p(x) |
| "x < y" | q(x, y) | "para todo x em $D_1$, existe y em $D_2$ tal que q(x, y)" | ∀x:∃y:q(x, y) |
| "x < y" | q(x, y) | "existe x em $D_1$ tal que para todo y em $D_2$, q(x, y)" | x:∀y:q(x, y) |

O **Valor Verdade** de uma função proposicional quantificada é determinado da seguinte forma:

- Se *p(x)* é verdadeira para **toda interpretação de x em D**
  então $\forall x : p(x)$ é verdadeira
  senão $\forall x : p(x)$ é falsa
- Se *p(x)* é verdadeira para **pelo menos uma interpretação de x em D**
  então $\exists x : p(x)$ é verdadeira
  senão $\exists x : p(x)$ é falsa

**Observação:** valor verdade de uma função proposicional depende do domínio *D* utilizado

- Se *D* é **finito**, com elementos $x_1, x_2, \ldots, x_n$,
  então $\forall x : p(x)$ **é equivalente a** $p(x_1) \wedge p(x_2) \wedge \ldots \wedge p(x_n)$ e
  $\exists x : p(x)$ **é equivalente a** $p(x_1) \vee p(x_2) \vee \ldots \vee p(x_n)$
- Se *D* é **vazio**, não contém elementos,
  então $\forall x : p(x)$ **é verdadeira** e
  $\exists x : p(x$ **é falsa**

## Para refletir

**1.** Seja $x \in D = \{1, 2, 3, 4\}$, $y \in S = \{-1, 0, 1, 2\}$, *p(x)* é *"x < 3"* e *q(y)* é *"y < 3"*. Verifique o valor verdade das seguintes proposições: $\forall x : p(x)$, $\forall y : q(y)$, $x : p(x)$, $\exists y : q(y)$.

**2.** Seja $x \in D$ o conjunto de matemáticos acima de três metros de altura, *p(x) é "x gosta de chocolate"*. Verifique o valor verdade das seguintes proposições: $\forall x : p(x)$, $\exists x : p(x)$.

## 2. Negação de funções proposicionais quantificadas

Funções proposicionais quantificadas com "*para todo*" e "*existe*" estão relacionadas através do conectivo de negação, da forma como se segue:

$\neg(\forall x : p(x)) \ \exists x : \neg p(x)$

$\neg(\exists x : p(x)) \leftrightarrow \forall x : \neg p(x)$

Exemplo:

$\neg(\forall x : (p(x) \rightarrow q(x))) \leftrightarrow \exists x : (p(x) \wedge \neg q(x))$

$\neg(\exists x : (p(x) \wedge q(x))) \leftrightarrow \forall x : (\neg p(x) \vee \neg q(x))$

### 2.1 Funções proposicionais quantificadas em linguagem natural

Em linguagem natural, as funções proposicionais quantificadas são frequentemente utilizadas de diversas formas, como mostrado nos exemplos a seguir:

- **"Existe um inteiro tal que seu quadrado é nove"**

  D = conjunto dos inteiros

  p(x) é "$x^2 = 9$"

  $\exists x : p(x)$

- **"Se f é diferenciável então f é contínua"**

  D = conjunto de funções

  p(f) é "f é diferenciável"

  q(f) é "f é contínua"

  $\forall f : (p(f) \rightarrow q(f))$

- **"Todo estudante de lógica entende quantificadores"**

  D = conjunto de estudantes

  p(x) é "x é estudante de lógica"

  q(x) é "x entende quantificadores"

  $\forall x : (p(x) \rightarrow q(x))$

- **"Alguns estudantes de lógica entendem quantificadores"**

  D = conjunto de estudantes

  p(x) é "x é estudante de lógica"

  q(x) é "x entende quantificadores"

  $\exists x : (p(x) \wedge q(x))$

**Observações:**

*"Todo p é um q"* representado por $\forall x : (p(x) \rightarrow q(x))$

*"Algum p é um q"* representado por $\exists x : (p(x) \wedge q(x))$

## 2.2 Negação de funções proposicionais quantificadas em português

A relação de negação existente entre os quantificadores de funções proposicionais quantificadas é frequentemente usada em linguagem natural na geração de sentenças logicamente equivalentes, como mostrado no exemplo abaixo.

- "Todo estudante de lógica entende quantificadores"

| | |
|---|---|
| **Representação em Lógica de Predicados** | $\forall x : (p(x) \rightarrow q(x))$ |
| **Negação da representação** | $\exists x : (p(x) \wedge \neg q(x))$ |
| **Significado da representação em português** | "Alguns estudantes de lógica não entendem quantificadores" |

### Para refleir

**1.** Encontre uma negação para cada uma das proposições:

a) Existe um inteiro x tal que 4 = x + 2.

b) Para todo inteiro x, 4 = x + 2.

c) Todo estudante gosta de lógica.

d) Alguns estudantes não gostam de lógica.

e) Nem um homem é uma ilha.

f) Todo mundo que estuda lógica gosta.

g) Toda pessoa tem uma mãe.

**2.** Seja *D* o conjunto dos números naturais (isto é, *D = {1, 2, 3, 4, ...}*), *p(x) é "x é par"*, *q(x) é "x é divisível por 3"* e *r(x) é "x é divisível por 4"*. Expresse em português, determine valores verdade e dê uma negação para cada uma das proposições.

a) $\forall x : p(x)$.

b) $\forall x : (p(x) \vee q(x))$.

c) $\forall : (p(x)$ □ $q(x))$.

d) $\forall : (p(x) \vee r(x))$.

e) $\forall x : (p(x) \wedge q(x))$.

f) $\exists x : r(x)$.

g) $\exists x : (p(x) \wedge q(x))$.

h) $\exists x : (p(x)$ □ $q(x))$.

i) $\exists x : (q(x)$ □ $q(x + 1))$.

## 3. Sentenças declarativas que envolvem mais de um quantificador

É importante destacar que uma dada função declarativa pode ser quantificada simultaneamente por mais de um quantificador.

Exemplos:

*"Para todo inteiro par n, existe um inteiro k tal que n = 2k"*

*"Para todo y em B, existe um x em A tal que y = f(x)"*

No entanto, é importante perceber que a ordem na qual os quantificadores estão dispostos interfere na interpretação lógica da sentença, como mostrado abaixo.

- Se $S = \{x_1, x_2, ..., x_n\}$ e $T = \{y_1, y_2, ..., y_m\}$
  então $\forall x : (\exists y : p(x,y))$ não é equivalente a $\exists y : (\forall x : p(x,y))$.

Exemplo:

- Se $S = T = \{1, 2\}$ e $p(x,y)$ é *"x = y"*

  então $\forall x : (\exists y : p(x,y))$ é uma proposição verdadeira e

  $\exists y : (\forall x : p(x,y))$ é uma proposição falsa

### Para refleir

**1.** Seja S = T = conjunto de todas as pessoas e p(x,y) é "y é mãe de x". Qual o valor verdade de $\forall x:(\exists y:p(x,y))$ e $\exists y:(\forall x:p(x,y))$?

**2.** Seja a proposição: "Para todo cachorro preto no sofá existe uma pulga no carpete tal que se o cachorro é preto então a pulga pica o cachorro".

a) Qual a representação da proposição em Linguagem Lógica de Predicados?

b) Qual a negação da proposição em Linguagem Lógica de Predicados?

c) Qual a negação da proposição em português?

## 4. Equivalências lógicas

Além da equivalência lógica dada pela negação entre os quantificadores "para todo" e "existe", outras equivalências existem, como apresentado abaixo:

- $\exists x : (\exists y : p(x,y)) \leftrightarrow \exists y : (\exists x : p(x,y))$
- $\forall x : (\forall y : p(x,y)) \leftrightarrow \forall y : (x : p(x,y))$
- $((\exists x : p(x)) \vee (\exists x : q(x))) \leftrightarrow \exists x : (p(x) \vee q(x))$
- $((\forall x : p(x)) \wedge (\forall x : q(x))) \leftrightarrow \forall x : (p(x) \wedge q(x))$

## 5. Implicações lógicas

Além dessas equivalências lógicas, algumas implicações lógicas podem ser desenhadas.

- $\exists y : (\forall x : p(x,y)) \rightarrow \forall x : (\exists y : p(x,y))$
- $((\forall x : p(x)) \vee (\forall x : q(x))) \rightarrow \forall x : (p(x) \vee q(x))$
- $\exists x : (p(x) \wedge q(x)) \rightarrow ((\exists x : p(x)) \wedge (x : q(x)))$
- $\forall x : (p(x) \rightarrow q(x)) \rightarrow ((\forall x : p(x)) \rightarrow (\forall x : q(x)))$

As **idéias e métodos de análise** utilizados para declarações envolvendo dois quantificadores podem ser estendidos para **três (ou mais) quantificadores.**

### Atividades de avaliação

**1.** Seja S = conjunto de inteiros, p(x) é "x é par" e q(x) é "x é impar".

Verifique as duas últimas equivalências lógicas acima.

**2.** Seja S = conjunto de inteiros, p(x) é "x é par" e q(x) é "x é impar".

Verifique as três últimas implicações lógicas acima.

**3.** Traduzir as seguintes sentenças em formas simbólicas, indicando escolhas apropriadas para os domínios:

a) Para todo inteiro par n existe um inteiro k tal que n = 2k.

b) Para toda linha l e todo ponto p não em l existe uma linha l' que passa por p que é paralela a l.

c) Para todo y em B existe um x em A tal que f(x) = y.

d) Para todo x em G existe um x' em G tal que xx' = e.

e) Se todo inteiro é impar então todo inteiro é par.

f) Todo mundo ama alguém alguma vez.

g) Para todo inteiro n existe outro inteiro maior que 2n.

h) A soma de dois inteiros pares é par.

**4.** Ache uma negação em Português para todas as proposições no exercício acima.

**5.** Considere que p(x,y) representa "x + 2 > y" e D o conjunto dos números naturais (D= {1, 2, 3,...}). Escreva em palavras e atribua valores verdade para

a) $\forall x : (\exists y : p(x,y))$.  d) $\exists x : (\exists y : p(x,y))$.

b) $\exists x : (y : p(x,y))$.  e) $\forall y : (\exists x : p(x,y))$.

c) $\forall x : (\forall y : p(x,y))$.  f) $\exists y : (\forall x : p(x,y))$.

**6.** Considere a seguinte proposição: "Para toda galinha no galinheiro e para toda cadeira na cozinha existe uma frigideira no armário tal que se o ovo da galinha está na frigideira então a galinha está a dois metros da cadeira".

a) Traduza em forma simbólica.

b) Expresse sua negação em símbolos e em Português.

Capítulo 5

# Representação do conhecimento e programação em lógica

## Introdução

A Programação em Lógica de Predicados é composta pelos seguintes elementos:

- Um **programa** é uma teoria formalizada em Linguagem Lógica de Predicados, isto é, uma coleção finita de fórmulas em Linguagem Lógica de Predicados;
- Uma **consulta** é qualquer fórmula em Linguagem Lógica de Predicados exprimindo as condições a serem satisfeitas por uma resposta correta;
- Uma **resposta** correta é um indivíduo do universo de discurso ***U***; e
- Um **método de busca** que pode ser utilizado para obtenção de respostas é a Resolução para Lógica de Predicados.

Exemplo:

| Programa em Lógica de Predicados | Consulta em Lógica de Predicados |
|---|---|
| $F_1$. programa(Soma.for, FORTRAN) | $F_6$. $\exists x$ : procedimental(x) |
| $F_2$. programa(Soma.pro, PROLOG) | |
| $F_3$. $\forall x$ : (programa(x, FORTRAN) $\rightarrow$ procedimental(x)) | |
| $F_4$. $\forall x$ : (programa(x, PASCAL) $\rightarrow$ procedimental(x)) | |
| $F_5$. $\forall x$ : (programa(x, PROLOG) $\rightarrow$ declarativo(x)) | |

## 1. Programa em Linguagem natural e sua representação em Linguagem Lógica de Predicados

1. "*Marcos era um homem*"
   *homem (Marcos)*
2. *"Marcos Nasceu em Pompéia"*
   *pompeano (Marcos)*
3. *"Todos os que nasceram em Pompéia eram romanos"*
   $\forall x : (Pompeano(x) \rightarrow Romano(x))$
4. *"César era um soberano"*
   *soberano (César)*
5. *"Todos os romanos eram leais a César ou então odiavam-no*
   $\forall x : (Romano(x) \rightarrow (leal\_a(x,César) \vee odeia(x,César)))$
6. *"Marcos tentou assassinar César"*
   *tentou_assassinar (Marcos,César)*
7. *"As pessoas só tentam assassinar soberanos aos quais não são leais"*
   $\forall x : (\forall y : ((pessoa(x) \wedge soberano(y) \wedge tentou\_assassinar(x,y)) \rightarrow \neg leal\_a(x,y)))$
8. *"Todo mundo é leal a alguém"*
   $\forall x : (\exists y : leal\_a(x,y))$
9. *"Todos os homens são pessoas"*
   $\forall x : (homem(x) \rightarrow pessoa(x))$

## 2. Consulta em Linguagem natural e sua representação em Linguagem Lógica de Predicados

10. *"Marcos era leal a César?"*
    *leal_a(Marcos,César)* ou
    $\neg leal\_a(Marcos,César)$

Exemplo de um **método de busca de respostas**: **raciocínio a partir do objetivo para trás** – a fim de responder à consulta, isto é, a fim de comprovar o objetivo, utilizamos "modus ponens" como fundamento e transformamos o objetivo principal (consulta) em um subobjetivo (ou, possivelmente, em um conjunto de subobjetivos) que possa, por sua vez, ser transformado e assim por diante, até todos os subobjetivos gerados serem satisfeitos (ou seja, até todas as variáveis geradas no processo de raciocínio serem substituídas por constantes apropriadas).

**Resposta**: Marcos não era leal a César.

$\neg leal\_a(Marcos,César)$

|  (7, substituição)

$pessoa(Marcos) \wedge$

$soberano(César) \wedge$

$tentou\_assassinar(Marcos,César)$

|  (4)

$pessoa(Marcos) \wedge$

$tentou\_assassinar(Marcos,César) \wedge$

|  (6)

$pessoa(Marcos)$

|  (9)

$homem(Marcos)$

|  (1)

**Aspectos importantes** que precisam ser consideradas no processo de conversão de frases em linguagem natural para Linguagem Lógica de Predicados:

**a)** Frases em linguagem natural são **ambíguas**, portanto, pode ser difícil escolher uma representação correta.

Exemplo:

**Declaração 5:**

"Todos os romanos eram leais a César ou então odiavam-no"

$\forall x$:(Romano(x) $\rightarrow$ (leal_a(x,César) $\vee$ odeia(x,César))) ou

$\forall x$:(Romano(x) $\rightarrow$ ((leal_a(x,César) $\vee$ odeia(x,César)) $\wedge$ $\neg$(leal_a(x,César) $\vee$ odeia(x,César)))

**b)** Para se usar um conjunto de declarações eficientemente, normalmente é necessário ter acesso a um outro conjunto de declarações que representam **fatos** que as pessoas consideram **óbvios** demais para serem declarados.

Exemplo:

**Declaração 9:**

*"Todos os homens são pessoas"*

$\forall x$:*(homem(x)* $\rightarrow$ *pessoa(x))*

**c)** A relação **elemento pertence conjunto** pode ser representada por uma fórmula atômica da forma *conjunto(elemento)*.

Exemplo:

**Declaração 2:** *"Marcos Nasceu em Pompéia"* - Elemento Marcos pertence ao conjunto dos pompeanos

*Pompeano(Marcos) - pertence(Marcos, Pompeanos)*

**Declaração 4:** *"César era um soberano"* - Elemento César pertence ao conjunto dos soberanos

*soberano(César) - pertence (César, Soberanos)*

**d)** A relação **conjunto1 contido conjunto2** pode ser representada por uma fórmula da forma $\forall x:(conjutno1(x) \rightarrow conjunto2(x))$.

Exemplo:

**Declaração 3:** *"Todos os que nasceram em Pompéia eram ro manos"* - Conjunto de pompeanos contido no de romanos

$\forall x:(Pompeano(x) \rightarrow Romano(x))$ - *contido(Pompeanos,Romanos)*

## Atividades de avaliação

**1.** Represente as seguintes sentenças por fbf's em lógica de predicados:

a) "Um sistema de computador é inteligente se pode realizar uma tarefa que, se realizada por um humano, requer inteligência".

b) "Uma fórmula cujo conectivo principal é uma → é equivalente a alguma fórmula cujo principal conectivo é um ∨".

c) "Se a entrada para o algoritmo da unificação é um conjunto de expressões unificáveis, a saída é o mgu (unificador mais geral); se a entrada é um conjunto de expressões não unificáveis, a saída é fracasso".

d) "Se não se pode dizer um fato a um programa, então o programa não pode aprender o fato.

**2.** Traduza as seguintes fbf's em Lógica de Predicados para o português.

a) ∀x : (duvida(x) → perde(x)).

b) ¬∃x : (homem_negócios(x) ∧ gosta(x,vida_noturna)).

c) ¬∀x : (brilhante(x) → ouro(x)).

d) ∀x : (∀y : (∀z : (mais_rápido(x,y) ∧ mais_rápido(y,z)) → mais_rápido (x,z))).

**3.** O que está errado com o seguinte argumento?

- *"Os homens estão amplamente distribuídos pela Terra".*
- *"Sócrates é um homem".*
- *"Portanto, Sócrates está amplamente distribuído pela Terra"*

**4.** Como é que os fatos representados pela sentença acima devem ser representados na lógica para que este problema não ocorra.

**5.** Expresse as seguintes sentenças em Lógica de Predicados. Use somente o objeto simbólico *Erasmus* e as relações simbólicas *homem(x)*, *mulher(x)*, *vegetariano(x)*, *Açougueiro(x)*, *Gosta(x,y)*, *=(x,y).*

a) Erasmus não gosta de nenhum vegetariano.

b) Erasmus não gosta de todo vegetariano.

c) Nenhum homem é açougueiro e vegetariano.

d) Todos os homens gostam de vegetarianos, exceto os açougueiros.

e) Alguns vegetarianos não gostam de todos os açougueiros.

f) Os únicos vegetarianos são homens que gostam de vegetarianos.

g) Homens só gostam de mulheres que não são vegetarianas.

h) Nenhuma mulher gosta de qualquer homem que não gosta de todos os vegetarianos.

i) Todo homem gosta de uma mulher diferente.

**6.** Traduza as seguintes fbf's em Lógica de Predicados para o português.

a) $\forall x : (duvida(x) \rightarrow perde(x))$.

b) $\neg\exists x : (homem\_negócios(x) \wedge gosta(x,vida\_noturna))$.

c) $\neg\forall x : (brilhante(x) \rightarrow ouro(x))$.

d) $\forall x : (\forall y : (\forall z(mais\_rápido(x,y) \wedge mais\_rápido(y,z)) \rightarrow mais\_rápido\ (x,z)))$.

Capítulo 6

# Funções e Predicados Computáveis e a Noção de Igualdade

## Introdução

Considerando que estamos interessados nos aspectos de construção de programas provadores automáticos, capazes de encontrar respostas para consultas, as funções computáveis são necessárias quando o programa precisar avaliar a verdade de predicados que possuam expressões funcionais como termos. Por exemplo:

*"2 + 3 é maior que 1"*

*maq(+(2, 3), 1))*

Neste caso, o programa provador deve chamar uma **função computável**, capaz de computar o valor da expressão *2+3* e, em seguida, avaliar o predicado *maq(5, 1)*.

No programa em lógica apresentado anteriormente, utilizamos **predicados isolados** (para diferenciar de predicados computáveis) para a representação de fatos simples, e uma combinação destes predicados para a representação de relações entre fatos. Por exemplo:

6. *"Marcos tentou assassinar César"*
   *tentou_assassinar(Marcos,César)*

9. *"Todos os homens são pessoas"*

   $\forall x : (homem(x) \rightarrow pessoa(x))$

A representação por meio de predicados isolados deve ser empregada quando o número de fatos não for muito grande. Se, por exemplo, desejarmos representar fatos que expressem relações do tipo *maior que*, em um conjunto composto por um número grande de elementos numéricos,

*"um é maior que zero"*
*maq(1, 0)*

*"dois é maior que um"*

*maq(2, 1)*

*"três é maior que 2"*

*maq(3, 2)*

*...*

Deveremos empregar um **predicado computável**. Assim, em vez do programa provador automático procurar pelo predicado explicitamente no conjunto de fórmulas disponíveis (ou tentar deduzir o predicado através de mais raciocínio) para a obtenção de respostas, ele deve invocar um procedimento e este avaliará o predicado.

A **noção de igualdade** permite que indivíduos iguais do universo de discurso sejam substituídos uns pelos outros sempre que este procedimento parecer útil durante o processo de busca de respostas. Por exemplo, a representação da sentença abaixo, utilizando a noção de igualdade,

"Estamos agora em 1998"
*agora = 1998*

indica que sempre que a constante *agora* aparecer como termo de algum predicado, ela poderá ser substituída pela constante *1998*.

## 1. Programa em Linguagem natural e sua representação em Linguagem Lógica de Predicados

1. "*Marcos era um homem*"
   *homem(Marcos)*
2. *"Marcos Nasceu em Pompéia"*
   *pompeano(Marcos)*
3. *"Marcos nasceu em 40 d.C."*
   *nasceu(Marcos,40)*
4. *"Todos os homens são mortais"*
   $\forall x : (homem(x) \rightarrow mortal(x))$
5. *"Todos os habitante de Pompéia morreram quando o vulcão entrou em erupção em 79 d.C."*
   $\forall x : (Pompeano(x) \rightarrow morreu(x,79))$
   *entrou_em_erupção(vulcão,79)*
6. *"Nenhum mortal vive mais que 150 anos"*
   $\forall x : (\forall t_1 : (\forall t_2 : ((mortal(x) \wedge nasceu(x,t_1) \wedge \mathbf{maq(-(}t_2,t_1),150)) \rightarrow morto(x,t_2))))$

7. *"Estamos agora em 1998"*
   *agora* = *1998*

8. *"Estar morto significa não estar vivo"*
   $\forall x : (\forall t : ((morto(x,t) \rightarrow \neg vivo(x,t)) \wedge (\neg vivo(x,t) \rightarrow morto(x,t)))$

9. *"Se alguém morre então ele está morto em todos os momentos posteriores"*
   $\forall x : (\forall t_1 : (\forall t_2 : ((\ morreu(x,t_1) \wedge \mathbf{maq}(\ t_2,t_1)) \rightarrow morto(x,t_2))))$

## 2. Consulta em Linguagem natural e sua representação em Linguagem Lógica de Predicados:

10. *"Marcos está vivo?"*
    *vivo(Marcos, agora)* ou $\neg$*vivo(Marcos, agora)*

## 3. Método de busca de respostas: Raciocínio a partir do objetivo para trás

**Resposta 1**: Marcos não está vivo (porque foi morto pelo vulcão).

*¬vivo(Marcos, agora)*

| (8, substituição)

*morto(Marcos, agora)*

| (9, substituição)

*morreu(Marcos, $t_1$) ∧*

*maq(agora, $t_1$)*

| (5, substituição)

*Pompeano(Marcos)*

*maq(agora, 79)*

| (2)

*maq(agora, 79)*

| (7)

*maq(1998, 79)*

| (calcular *maq*)

nil

**Resposta 2**: Marcos não está vivo (senão estaria com mais de 150 anos).

$\neg vivo(Marcos, agora)$

| (8, substituição)

$morto(Marcos, agora)$

| (6, substituição)

$mortal(Marcos) \wedge$
$nasceu(Marcos, t_1) \wedge$
$maq(-(agora, t_1), 150)$

| (4, substituição)

$homem(Marcos) \wedge$
$nasceu(Marcos, t_1) \wedge$
$maq(-(agora, t_1), 150)$

| (1)

$nasceu(Marcos, t_1) \wedge$
$maq(-(agora, t_1), 150)$

| (3, substituição)

$maq(-(agora, 40), 150)$

| (7, substituição)

$maq(-(1998, 40), 150)$

| (computar -)

$maq(1958, 150)$

| (computar *maq*)

nil

**Observações:**

- A representação de fatos que adquirem gradualmente com a experiência pode diminuir o número de etapas para a consulta ser respondida.
- Um bom método de busca de respostas deve ser capaz de:
  - Determinar que existe um casamento entre dois predicados

    Ex: *morto (Marcos, agora)* casa com $morto(x, t_2)$,
  - Garantir substituições uniformes em toda a prova r
  - Aplicar *modus ponens*.
  - Do ponto de vista computacional, precisamos de um método de busca de respostas capaz de executar em uma única operação a série de processos envolvidos no raciocínio com declarações em Lógica de Predicados.

## Atividades de avaliação

**1.** Usando o primeiro programa em lógica desta seção responda à consulta: *"Marcos odiava César?"*

**2.** Considere as seguintes sentenças:

- *"João gosta de todo tipo de comida".*
- *"Maçãs são comidas".*
- *"Frango é comida"*
- *"Qualquer coisa que alguém coma e que não cause sua morte é comida".*
- *"Paulo come amendoim e está vivo".*
- *"Susana come tudo o que Paulo come"*

Traduza as sentenças em fbf's em Lógica de Predicados e responda a consulta abaixo, usando o raciocínio para trás:

- *"João gosta de amendoim?"*

# Parte 3

# Resolução

Capítulo 7

# Conversão para Forma Clausal

## Objetivos

- Conhecer alguns dos principais algoritmos que podem ser utilizados durante o processo de prova automática de argumentos e de programação em Lógica de Predicados.
- Aplicar o algoritmo de prova automática da Resolução e seus algoritmos auxiliares na obtenção de respostas para consultas realizadas a programas em Lógica de Predicados.

## Introdução

A Resolução é um método de decisão que, em uma única operação, executa a série de rocessos envolvidos com declarações em Linguagem Lógica de Predicados.

Esse método opera em declarações que foram convertidas em uma forma padrão conveniente: a **Forma Clausal**.

As Respostas são obtidas por meio de um **processo de prova por refutação**.

Definições:

(a) Uma fórmula P é uma **Conjunção** se e somente se, omitindo-se os parênteses, for da forma $P_1 \wedge P_2 \wedge ... \wedge P_n$.

(b) Uma fórmula P é uma **Disjunção** se e somente se, omitindo-se os parênteses, for da forma $P_1 \vee P_2 \vee ... \vee P_n$.

(c) Uma fórmula P está em **Forma Normal Prenex** se e somente se P for da forma Q(M) onde Q, o prefixo de P, é uma cadeia de quantificadores e M, a matriz de P, é uma fórmula sem ocorrência de quantificadores.

(d) Uma fórmula P está em **Forma Normal Conjuntiva** se e somente se estiver em Forma Normal Prenex e sua matriz for uma conjunção de disjunções de fórmulas atômicas, negadas ou não.

(e) Uma fórmula P está em **Forma Clausal** se e somente se estiver em Forma Normal Conjuntiva mas sem nenhuma instância do conectivo $\wedge$.

Exemplo:

- Fórmula em Forma Normal Prenex: $\forall x : (\exists y : (p(x, y) \rightarrow (q(x) \wedge q(y))))$

  Prefixo: $\forall x : \exists y:$

  Matriz: $p(x, y) \rightarrow (q(x) \wedge q(y))$
- Fórmula em Forma Normal Conjuntiva: $\forall x : (\exists y : ((\neg p(x, y) \vee q(x)) \ (\neg p(x,y) \vee \ q(y))))$
- Fórmula na Forma Clausal

  Cláusula1: $\neg p(x, F(x)) \vee q(x)$

  Cláusula2: $\neg p(x, F(x)) \vee q(F(x))$

## 1. Algoritmo Conversão para forma Clausal

Para que a resolução funcione, primeiro, precisamos converte cada fórmula do programa em Forma Normal Conjuntiva e, em seguida, dividir cada expressão resultante em um conjunto de cláusulas.

A partir do algoritmo de conversão, considerando uma fórmula em Linguagem Lógica de Predicados e a execução dos passos abaixo sobre a fórmula, obteremos um conjunto de cláusulas equivalente a fórmula original.

**1.** Elimine $\rightarrow$, usando:

$(p \rightarrow q) \leftrightarrow (\neg p \vee q)$. - Tautologia 12.a).

**2.** Reduza o escopo de cada $\neg$ a um único termo usando:

$\neg\neg p \leftrightarrow p$, - Tautologia 5,

$\neg(p \wedge q) \leftrightarrow (\neg p \vee \neg q)$ e- Tautologia 10.a),

$\neg(p \wedge q) \leftrightarrow (\neg p \vee \neg q)$. - Tautologia 10.b);

e as correspondências padrões entre quantificadores:

$\neg(\forall x : p(x)) \leftrightarrow \exists x : \neg p(x)$ e

$\neg(\exists x : p(x)) \leftrightarrow \forall x : \neg p(x)$.

**3.** Padronize as variáveis para que cada quantificador fique vinculado a uma única variável, por exemplo:

$(\forall x : p(x)) \vee (\forall x : q(x))$

poderia ser convertida em

$(\forall x : p(x)) \vee (\forall y : q(y))$.

**4.** Coloque a fórmula na Forma Normal Prenex.

**5.** Elimine os quantificadores existenciais, ou seja:

$\exists y : presidente(y)$

pode ser transformada em $presidente(S_1)$,

e

$\forall x : \exists y : pai\_de(y,x)$

pode ser transformada em $\forall x : pai\_de(S_2(x),x)$

onde:

$S_1$ - é uma função sem argumentos, que de algum modo produz um valor que satisfaz presidente;

$S_2$ - é uma função com um argumento, $x$, que de algum modo produz para cada $x$ (em $D$) um valor que satisfaz pai_de.

**6.** Elimine o prefixo.

**7.** Converta a matriz em uma conjunção de disjunções, usando:

$(p \vee (q \wedge r)) \leftrightarrow ((p \vee q) \wedge (p \vee r))$. - Tautologia 8.b)

**8.** Crie uma cláusula separada que corresponda a cada conjunção:

$c_1$: $p \vee q$

e

$c_2$: $p \vee r$.

**9.** Padronize distintamente as variáveis do conjunto de cláusulas gerados no Passo 8. Esta transformação está baseada no fato que:

$(\forall x : (p(x) \wedge q(x))) \leftrightarrow ((\forall x : p(x)) \wedge (\forall x : q(x)))$.

Veja os exemplos a seguir:

## 2. Converter a fórmula abaixo para Forma Clausal

$\forall x : ((Romano(x) \wedge conhece(x, Marcos)) \rightarrow (odeia(x, César) \vee$
$(\forall y : ((\exists z : odeia(y, z)) \rightarrow acha\_louco(x, y)))))$

**1.** $\forall x : (\neg(Romano(x) \wedge conhece(x, Marcos)) \vee odeia(x, César) \vee (\forall y : (\neg(\exists z : odeia(y, z)) \vee acha\_louco(x, y))))$

**2.** $\forall x : (\neg Romano(x) \vee \neg conhece(x, Marcos) \vee odeia(x, César) \vee (\forall y : ((\forall z : \neg odeia(y, z)) \vee acha\_louco(x, y))))$

**3.** $\forall x : (Romano(x) \vee \neg conhece(x, Marcos) \vee odeia(x, César) \vee (\forall y : ((\forall z : \neg odeia(y, z)) \vee acha\_louco(x, y))))$

**4.** $\forall x : (y : (\forall z : (\neg Romano(x) \vee \neg conhece(x, Marcos) \vee odeia(x, César) \vee \neg odeia(y, z) \vee acha\_louco(x, y))))$

**5.** $\forall x : (y : (\forall z : (\neg Romano(x) \vee \neg conhece(x, Marcos) \vee odeia(x, César) \vee \neg odeia(y, z) \vee acha\_louco(x, y))))$

**6.** $\neg Romano(x) \vee \neg conhece(x, Marcos) \vee odeia(x, César) \vee \neg odeia(y, z) \vee acha\_louco(x, y)$

**7.** $\neg Romano(x) \vee \neg conhece(x, Marcos) \vee odeia(x, César) \vee \neg odeia(y, z) \vee acha\_louco(x, y)$

**8.** $\neg Romano(x) \vee \neg conhece(x, Marcos) \vee odeia(x, César) \vee \neg odeia(y, z) \vee acha\_louco(x, y)$

**9.** $\neg Romano(x) \vee \neg conhece(x, Marcos) \vee odeia(x, César) \vee \neg odeia(y, z) \vee acha\_louco(x, y)$

## 3. Aplicação do algoritmo ao programa que nos fala sobre o mundo de Marcos e César

**1.** $homem(Marcos)$
homem(Marcos)

**2.** $Pompeano(Marcos)$
Pompeano(Marcos)

**3.** $\forall x : (Pompeano(x) \rightarrow Romano(x))$
**3.1** $\forall x : (\neg Pompeano(x) \vee Romano(x))$
**3.6** $\neg Pompeano(x) \vee Romano(x)$
**3.9** $\neg Pompeano(x_1) \vee Romano(x_1)$

**4.** $soberano(César)$
soberano(César)

**5.** $\forall x : (Romano(x) \rightarrow (leal\_a(x, César) \vee odeia(x, César)))$
**5.1** $\forall x : (\neg Romano(x) \vee leal\_a(x, César) \vee odeia(x, César))$
**5.6** $\neg Romano(x) \vee leal\_a(x, César) \vee odeia(x, César)$
**5.9** $\neg Romano(x_2) \vee leal\_a(x_2, César) \vee odeia(x_2, César)$

**6.** *tentou_assassinar(Marcos, César)*
tentou_assassinar(Marcos, César)

**7.** $\forall x : (\forall y : ((pessoa(x) \wedge soberano(y) \wedge tentou\_assassinar(x, y)) \rightarrow \neg leal\_a(x, César)))$
**7.1** $\forall x : (\forall y : (\neg(pessoa(x) \wedge soberano(y) \wedge tentou\_assassinar(x, y)) \vee \neg leal\_a(x, César)))$
**7.2** $\forall x : (\forall y : (\neg pessoa(x) \vee \neg soberano(y) \vee \neg tentou\_assassinar(x, y) \vee \neg leal\_a(x, César)))$
**7.6** $\neg pessoa(x) \vee \neg soberano(y) \vee \neg tentou\_assassinar(x, y) \vee \neg leal\_a(x, César)$
**7.9** $\neg pessoa(x_3) \vee \neg soberano(y) \vee \neg tentou\_assassinar(x_3, y) \vee \neg leal\_a(x_3, César)$

**8.** $\forall x : (\exists y : leal\_a(x, y))$
**8.5** $\forall x : (leal\_a(x, S_1(x)))$
**8.6** $leal\_a(x, S_1(x))$
**8.9** $leal\_a(x_4, S_1(x_4))$

**9.** $\forall x : (homem(x) \rightarrow pessoa(x))$
**9.1** $\forall x : (\neg homem(x) \vee pessoa(x))$
**9.6** $\neg homem(x) \vee pessoa(x)$
**9.9** $\neg homem(x_5) \vee pessoa(x_5)$

## 4. Novo programa

**Cláusula1:** *homem(Marcos)*

**Cláusula2:** *Pompeano(Marcos)*

**Cláusula3:** $\neg Pompeano(x_1) \vee Romano(x_1)$

**Cláusula4:** *soberano(César)*

**Cláusula5:** $\neg Romano(x_2) \vee leal\_a(x_2, César) \vee odeia(x_2, César)$

**Cláusula6:** *tentou_assassinar(Marcos, César)*

**Cláusula7:** $\neg pessoa(x_3) \vee \neg soberano(y) \vee \neg tentou\_assassinar(x_3, y) \vee \neg leal\_a(x_3, César)$

**Cláusula8:** $leal\_a(x_4, S_1(x_4))$

**Cláusula9:** $\neg homem(x_5) \vee pessoa(x_5)$

## Atividades de avaliação

**1.** Colocar em forma normal conjuntiva a seguinte fórmula:

$\forall x : (cidade(x) \rightarrow (\exists y : guarda(x,y) \wedge (\forall z : ((cão(z) \wedge vive(x, z)) \rightarrow mordido(y, z)))))$.

Em seguida, dar três passos adicionais que completarão o processo de conversão da fórmula original em uma conjunção de cláusulas, útil no método da resolução:

- Eliminar quantificadores universais,
- Eliminar símbolos de conjunção e
- Rebatizar variáveis de forma que um certo termo do tipo variável não apareça em mais que uma cláusula.

**2.** Converta para forma clausal as seguintes fórmulas:

a) $\forall x : (\forall y : (P(x, y) \rightarrow Q(x, y)))$.

b) $\forall x : (\forall y : (\neg Q(x, y) \rightarrow \neg P(x, y)))$.

c) $\forall x : (\forall y : (P(x, y) \rightarrow (Q(x, y) \rightarrow R(x, y))))$.

d) $\forall x : (\forall y : (P(x, y) \wedge (Q(x, y) \rightarrow R(x, y))))$.

e) $\forall x : (\forall y : (P(x, y) \rightarrow (Q(x, y) \vee R(x, y))))$.

f) $\forall x : (\forall y : (P(x, y) \rightarrow (Q(x, y) \wedge R(x, y))))$.

g) $\forall x : (\forall y : ((P(x, y) \vee Q(x, y))) \rightarrow R(x, y)))$.

h) $\forall x : (\exists y : (P(x, y) \rightarrow Q(x, y)))$.

i) $\neg(\forall x : (\exists y : (P(x, y) \rightarrow Q(x, y))))$.

j) $\neg(\forall x : P(x)) \rightarrow (\exists x : P(x))$.

k) $\forall x : (P(x) \rightarrow P(x))$.

l) $\neg(\forall x : P(x)) \rightarrow (\exists x : \neg P(x))$.

# Capítulo 8

# Algoritmo da Unificação

## Introdução

Na Resolução para Lógica Proposicional é fácil para um programa provador automático obter uma cláusula-resolvente a partir de duas cláusulas-pais, contendo símbolos proposicionais complementares causadores de uma contradição. Por exemplo:

$$\underbrace{\neg \boldsymbol{p} \vee q \vee r \qquad\qquad \boldsymbol{p} \vee s}_{q \vee r \vee s.}$$

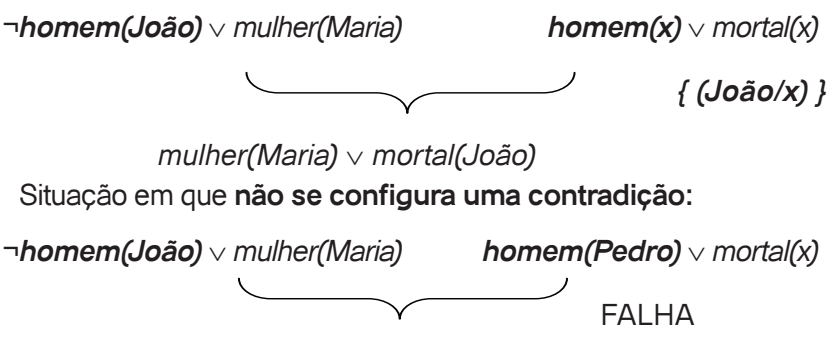
No exemplo, o programa procurou uma contradição, obteve a cláusula-resolvente como uma disjunção das cláusulas-pais e, em seguida, eliminou a contradição, eliminando os símbolos complementares $\neg p$ e $p$. Esta facilidade decorre de um processo de casamento simples, proporcionado pela presença de simples **símbolos proposicionais**.

Na Resolução para a Lógica de Predicados o processo de casamento entre **fórmulas atômicas** torna-se mais complicado, pois os argumentos dos predicados que compõem estas fórmulas também devem ser comparados. Neste caso, podem ocorrer diversos tipos de situações. Por exemplo:

- Situação em que **se configura uma contradição** sem substituição de variáveis:

$$\underbrace{\neg \textbf{\textit{homem(João)}} \vee \textit{mulher(Maria)} \qquad \textbf{\textit{homem(João)}} \vee \textit{mortal(x)}}_{\textit{mulher(Maria)} \vee \textit{mortal(x)}} \qquad \{\ \}$$

- Situação em que **se configura uma contradição** com substituição de variáveis:




- Situação em que **não se configura uma contradição:**




Para auxiliar o programa provador automático na busca de contradições, podemos implementar certo algoritmo denominado **Algoritmo da Unificação**. Este algoritmo descreve uma função recursiva que compara duas fórmulas atômicas e retorna, se existir, um conjunto de substituições que tornem as fórmulas idênticas.

## O Algoritmo

**Unificar(*L1, L2*)**
**1.** Se *L1* ou *L2* é variável ou constante
  então se *L1* e *L2* são idênticos
    então **retorne *{ }*.**
    senão se *L1* for variável
      então se *L1* ocorre em *L2*
        então **retorne *FALHA*.**
        senão **retorne *{ (L2/L1) }*.**
      senão se *L2* for variável
    então se *L2* ocorre em *L1*
      então **retorne *FALHA*.**
      senão **retorne *{ (L1/L2) }*.**
    senão **retorne *FALHA*.**
**2.** Se os símbolos predicativos em *L1* e *L2* não são idênticos
  então **retorne *FALHA*.**
**3.** Se *L1* e *L2* têm número de termos diferentes
  então **retorne *FALHA*.**
**4.** Faça *SUBST* := *{ }*.

**5.** Para i = 1, ..., número de termos em *L1*:

(a) Chame Unificar com o i-ésimo argumento de *L1* e o i-ésimo ar gumento de *L2*, colocando o resultado em *S*.

(b) Se *S* = *FALHA*

então **retorne *FALHA.***

(c) Se *S* ≠ *{ }*

então: aplique *S* ao restante de *L1* e *L2* e

faça *SUBST* := Concatena( *SUBST, S*).

**6. Retorne *SUBST.***.

**Observação sobre a função Concatena(*Lista1, Lista2*)**:

A função Concatena recebe como argumentos duas listas de substituições, armazenadas em *Lista1* e *Lista2*, e retorna uma terceira lista resultante da concatenação das duas primeiras. Por exemplo:

*SUBST* := Concatena(*{ (Marcos/x) }, { (z/y) }*)

*SUBST* := *{ (Marcos/x), (z/y) }*

*SUBST* := Concatena(*{ (z/y) }, { (Marcos/x) }*)

*SUBST* := *{ (z/y), (Marcos/x) }*

*SUBST* := Concatena( *{ }, { (Marcos/x), (z/y) }*)

*SUBST* := *{ (Marcos/x), (z/y) }*

*SUBST* := Concatena(*{ (Marcos/x), (z/y) }, { }*)

*SUBST* := *{ (Marcos/x), (z/y) }*

Exemplos:

Unificar(***homem(João), homem(João)***)

L1 := homem(João)

L2 := homem(João)

**4.** *SUBST* := *{ }* .

**5.** i := 1

(a) Unificar(João, João)

*S* := *{ }*.

**6.** Retornar { }.

Unificar(***homem(João), homem(x)***)

L1 := homem(João)

L2 := homem(x)

**4.** *SUBST := { }.*

**5.** i := 1

(a) Unificar(***João, x***)

S := { ( João/x) }.

(c) *L1 := homem(João)*

L2 := homem(x)

SUBST := { (João/x) }.

**6.** Retornar *{ (João/x) }.*

Unificar(***homem(João), homem(Pedro)***)

L1 := homem(João)

L2 := homem(Pedro)

**4.** *SUBST := { }.*

**5.** i := 1

(a) Unificar(João, Pedro)

S := FALHA.

(b) retornar FALHA.

Unificar(***odeia(Marcos,z), odeia(x,y)***)

L1 := odeia(Marcos, z)

L2 := odeia(x,y)

**4.** *SUBST := { }.*

**5.** i := 1

(a) Unificar(Marcos, x)

S := { ( Marcos/x)}

(c) *L1 := odeia(Marcos,z)*

*L2 := odeia(x,y)*

*SUBST := { (Marcos/x) }*

i := 2

(a) Unificar(z, y)

S := { ( y/z) }.

(c) *L1 := odeia(Marcos,z)*

L2 = odeia(x,y)

SUBST := { (Marcos/x), (y/z) }.

**6.** Retornar { (y/z), (Marcos/x) }.

Unificar(***f(x,x), f(g(x),x)***)

L1 := f(x,x)

L2 := f(g(x),x)

**4.** *SUBST := { }*.

**5.** i := 1

(a) Unificar(x, g(x))

S := FALHA.

(b) retornar FALHA.

**Observação**

Este último exemplo descreve um tipo de situação em que o Algoritmo da Unificação tenta unificar uma expressão funcional *g(x)*, envolvendo a variável *x*, com a própria variável *x*. Conforme se pode observar, neste caso, o Algoritmo *retorna* ***FALHA***, pois se ele aceitasse *g(x)* como substituição para *x*, *S := { (g(x)/x) }*, então quando ele fosse substituir *x* por *g(x)* no restante da fórmula atômica *L1* (Passo 5.(c)), ele estaria causando uma recursão infinita, ou seja:

*L1 := f(x,g(g(g(g... ;*

já que não seria possível eliminar *x de L1*.

## Atividades de avaliação

**1.** Utilize o Algoritmo da Unificação para produzir substituições apropriadas para os pares de fórmulas atômicas abaixo. Caso possível, ou seja, caso as fórmulas sejam unificáveis, expresse o retorno do Algoritmo. Caso não possível, ou seja, caso as fórmulas não sejam unificáveis, explique porque.

a) *cor(lolo,amarelo)* *cor(x,y)*

b) *cor(lolo,amarelo)* *cor(x,x)*

c) *cor(chapéu(carteiro),azul)* *cor(chapéu(y),x)*

d) *R(F(x),bbb)* *R(y,z)*

e) *R(F(y),x)* *R(x,F(bbb))*

f) *R(F(y),y,x)* *R(x,F(z),F(w))*

g) *ama(x,y)* *ama(y,x)*

Capítulo 9

# Algoritmo da Resolução

## Introdução

Basicamente, os passos que compõem o **algoritmo da Resolução para a Lógica de Predicados** são os mesmos realizados para a Lógica Proposicional. As diferenças estão nos algoritmos de Conversão para Forma Clausal e, no caso da Lógica de Predicados, a necessidade de se ter em mãos um procedimento de unificação, que possibilite verificar o casamento de duas fórmulas atômicas.

**Seja:** ***P*** = ***Programa*** = *{ Fórmulas em Lógica de Predicados } e*
***C*** = ***Consulta*** *a ser respondida pelo Algoritmo da Resolução = Fórmu la em Lógica de Predicados*

**1.** Converter Fórmulas de ***P*** para Forma Clausal.

**2.** Negar ***C***. Converter ¬***C*** para Forma Clausal e acrescentar resultado ao novo ***P***, obtido no Passo 1.

**3.** Repetir {

a) Selecionar duas ***cláusulas-pais, C1*** e ***C2***, tal que:

***C1*** *contém fórmula atômica* ***f***,

***C2*** contém *fórmula* ¬***f'***, e

***f*** *unifica com* ***f'***;

b) Obter:

***resolvente*** = *disjunção de todos as fórmulas atômicas de* ***C1*** e ***C2*** com a eliminação de ***f*** e ¬***f'***, realizando-se as substituições apropriadas na disjunção resultante*;*

c) Se ***resolvente*** ≠ **nil**

então acrescentar ***resolvente*** a ***P***.

senão ***Resposta*** foi encontrada.

} até uma ***Resposta*** ser encontrada, ou até não ser possível nem um progresso, ou até uma quantidade pré-determinada de repetições terem sido realizadas.

**4.** Responder.

Conforme você pode perceber, o Algoritmo da Resolução para a Lógica de Predicados, primeiramente, chama várias vezes o algoritmo de Conversão

para a Forma Clausal no Passo 1 e uma vez no Passo 2; posteriormente, chama, sempre que necessário, o algoritmo da Unificação em cada repetição no Passo 3.

Veja outros exemplos:

$P = \{F_1.\ homem(Marcos)\ ,$

$F_2.\ Pompeano(Marcos)\ ,$

$F_3.\ \forall x : (Pompeano(x) \rightarrow Romano(x))\ ,$

$F_4.\ soberano(César)\ ,$

$F_5.\ \forall x : (Romano(x) \rightarrow (leal\_a(x, César) \vee odeia(x, César)))\ ,$

$F_6.\ tentou\_assassinar(Marcos, César)\ ,$

$F_7.\ \forall x : (\forall y : ((pessoa(x) \wedge soberano(y) \wedge tentou\_assassinar(x, y)) \rightarrow \neg leal\_a(x,y)))\ ,$

$F_8.\ \forall x : (\exists y : leal\_a(x,y))\ ,$

$F_9.\ \forall x : (homem(x) \rightarrow pessoa(x))$

$\}$

$C = odeia(Marcos, César)$

**1.**

$P = \{\ C_1.\ homem(Marcos)\ ,$

$C_2.\ Pompeano(Marcos)\ ,$

$C_3.\ \neg Pompeano(x_1) \vee Romano(x_1)\ ,$

$C_4.\ soberano(César)\ ,$

$C_5.\ \neg Romano(x_2)\ \ leal\_a(x_2,César) \vee odeia(x_2,César)\ ,$

$C_6.\ tentou\_assassinar(Marcos,César)\ ,$

$C_7.\ \neg pessoa(x_3) \vee \neg soberano(y) \vee \neg tentou\_assassinar(x_3,y) \vee \neg leal\_a(x_3,y),$

$C_8.\ leal\_a(x_4,S_1(x_4))\ ,$

$C_9$. $\neg homem(x_5) \vee pessoa(x_5)$

}.

**2.**

$\neg C = \neg odeia(Marcos,César).$

$P = \{ C_1.\ homem(Marcos)$ ,

$C_2.\ Pompeano(Marcos)$ ,

$C_3.\ \neg Pompeano(x_1) \vee Romano(x_1)$ ,

$C_4.\ soberano(César)$ ,

$C_5.\ \neg Romano(x_2) \vee leal\_a(x_2, César) \vee odeia(x_2, César)$ ,

$C_6.\ tentou\_assassinar(Marcos, César)$ ,

$C_7.\ \neg pessoa(x_3) \vee \neg soberano(y) \vee \neg tentou\_assassinar(x_3, y) \vee$

$\neg leal\_a(x_3, y)$,

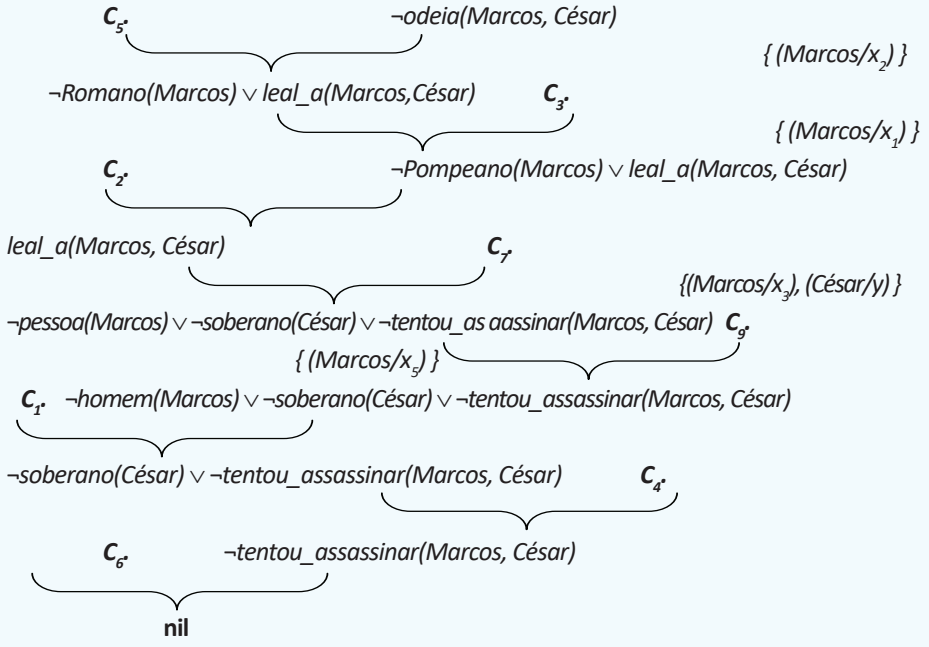
$C_8.\ leal\_a(x_4, S_1(x_4))$ ,

$C_9.\ \neg homem(x_5) \vee pessoa(x_5)$ ,

$C_{10}.\ \neg odeia(Marcos, César)$

}.

**3.**




**4. *Resposta = 'sim'*.**

**Heurísticas** que podem ser utilizadas para acelerar o processo de busca de uma resposta:

**a)** Selecionar Cláusulas-pais que contenham fórmulas atômicas complementares.

**b)** Eliminar resolventes que sejam:

- tautologias,
- subordinados a outras cláusulas existentes ($p \vee q$ está subordinado a $p$).

**c)** Iniciar processo de resolução resolvendo a cláusula que se deseja refutar (consulta negada em forma clausal) e, sempre que possível, continuar resolvendo os resolventes gerados a partir desta.

**d)** Sempre que possível, resolver cláusulas-pais que contenham uma única fórmula atômica.

## 1. Situações nas quais a Resolução pode detectar que não existe contradição

- No programa não existe uma cláusula que contenha uma fórmula atômica complementar a alguma fórmula atômica da consulta negada em forma clausal, por exemplo:

$P = \{ C_1.\ homem(Marcos),$
$C_2.\ Pompeano(Marcos),$
$C_3.\ \neg Pompeano(x_1) \vee Romano(x_1),$
$C_4.\ soberano(César),$
$C_5.\ \neg Romano(x_2)\ \ leal\_a(x_2, César) \vee odeia(x_2, César),$
$C_6.\ tentou\_assassinar(Marcos, César),$
$C_7.\ \neg pessoa(x_3) \vee \neg soberano(y) \vee \neg tentou\_assassinar(x_3, y) \vee \neg leal\_a(x_3, y),$
$C_8.\ leal\_a(x_4, S_1(x_4)),$
$C_9.\ \neg homem(x_5) \vee pessoa(x_5)$
$\}.$

$\neg C = odeia(Marcos, César).$

- Quando a Resolução partir de uma certa consulta negada e gerar um tipo de situação semelhante a situação descrita acima, por exemplo:

$$\neg C = \neg leal\_a(Marcos, César).$$

**Resolução**:

$C_5.$ $\neg leal\_a(Marcos, César).$
$\{ (Marcos/x_2) \}$
$\neg Romano(Marcos) \vee odeia(Marcos, César)$ $C_3.$
$\{ (Marcos/x_1) \}$
$C_2.$ $\neg Pompeano(Marcos) \vee odeia(Marcos, César)$
$odeia(Marcos, César)$ $?$

## 2. Resolução lidando com Funções e Predicados Computáveis e a Noção de Igualdade

$P = \{$ $F_1.\ homem(Marcos),$
$F_2.\ Pompeano(Marcos),$
$F_3.\ nasceu(Marcos, 40),$
$F_4.\ \forall x:(homem(x) \rightarrow mortal(x)),$
$F_5.\ entrou\_em\_erupção(vulcão, 79) \wedge \forall x:(Pompeano(x) \rightarrow morreu(x, 79)),$
$F_6.\ \forall x:(\forall t_1:(\forall t_2:((mortal(x) \wedge nasceu(x, t_1) \wedge \mathbf{maq}(-(t_2, t_1), 150)) \rightarrow morto(x, t_2))))$
$F_7.\ agora = 1998,$
$F_8 \forall x:(\forall t:(morto(x, t) \rightarrow \neg vivo(x, t)),$
$F_9.\forall x:(\forall t:(\neg morto(x, t) \rightarrow vivo(x, t)),$
$F_{10}.\ \forall x:(\forall t_1:(\forall t_2:((morreu(x, t_1) \wedge \mathbf{maq}(t_2, t_1)) \rightarrow morto(x, t_2))))$
$\}$

$C = \neg vivo(Marcos, agora)$

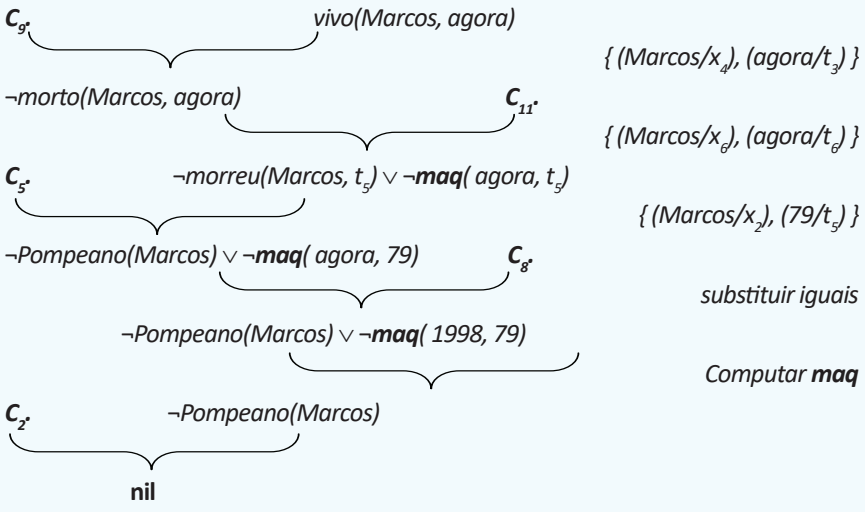
**1.**

$P = \{ \quad C_1.\ homem(Marcos)$,
$C_2.\ Pompeano(Marcos)$,
$C_3.\ nasceu(Marcos, 40)$,
$C_4.\ \neg homem(x_1) \vee mortal(x_1)$,
$C_5.\ \neg Pompeano(x_2)\ \ morreu(x, 79)$,
$C_6.\ entrou\_em\_erupção(vulcão, 79)$,
$C_7.\ \neg mortal(x) \vee \neg\ nasceu(x, t_1) \vee \neg\ \mathbf{maq}(-(t_2, t_1), 150) \vee\ morto(x, t_2)$
$C_8.\ agora = 1998$,
$C_9.\ \neg morto(x, t) \vee\ \neg vivo(x, t)$,
$C_{10}.\ morto(x, t) \vee vivo(x, t)$,
$C_{11}.\ \neg morreu(x, t_1) \vee \neg\mathbf{maq}(t_2, t_1) \vee morto(x, t_2)$
$\}$.

**2.**

$\neg C = vivo(Marcos,agora)$.
$P = \{ \quad C_1.\ homem(Marcos)$,
$C_2.\ Pompeano(Marcos)$,
$C_3.\ nasceu(Marcos, 40)$,
$C_4.\ \neg homem(x_1) \vee mortal(x_1)$,
$C_5.\ \neg Pompeano(x_2)\ \ morreu(x_2, 79)$,
$C_6.\ entrou\_em\_erupção(vulcão, 79)$,
$C_7.\ \neg mortal(x_3) \vee \neg\ nasceu(x_3, t_1) \vee \neg\ \mathbf{maq}(-(t_2, t_1), 150) \vee\ morto(x_3, t_2)$,
$C_8.\ agora = 1998$,
$C_9.\ \neg morto(x_4, t_3) \vee\ \neg vivo(x_4, t_3)$,
$C_{10}.\ morto(x_5, t_4) \vee vivo(x_5, t_4)$,
$C_{11}.\ \neg morreu(x_6, t_5) \vee \neg\mathbf{maq}(t_6, t_5) \vee\ morto(x_6, t_6)$
$C_{12}.\ vivo(Marcos, agora)$
$\}$

**3.**




**4. *Resposta = 'sim'***.

**Formas básicas** em que as **consultas** em Linguagem Natural podem aparecer e tipos de **respostas** correspondentes:

a) Perguntas que exigem respostas do **tipo *sim-não***:

- Consulta em Linguagem Natural: *"Marcos está vivo ?"*
- Consulta em Lógica de Predicados: *¬morto(Marcos,agora)*
- Respostas: *'sim"* ou *'não'*.

b) Perguntas que exigem respostas do **tipo *preenchimento de lacunas***:

- Consulta em Linguagem Natural: *"Quem tentou assassinar um soberano ?"*
- Consulta em Lógica de Predicados: $\exists x : (\exists y : ( tentou\_assassinar(x, y) \wedge soberano(y)))$
- Respostas: *substituição de indivíduos do universo de discurso no lugar de x, digamos ($x_0$ / x), e de y, digamos ($y_0$ / y), tal que tentou_assassinar($x_0$, $y_0$) ∧ soberano($y_0$) é verdadeiro.*

Exemplos:

$P = \{ F_1.\ homem(Marcos),$
$F_2.\ Pompeano(Marcos),$
$F_3.\ nasceu(Marcos, 40),$
$F_4.\ \forall x:(homem(x) \rightarrow mortal(x)),$
$F_5.\ entrou\_em\_erupção(vulcão, 79) \wedge \forall x : (Pompeano(x) \rightarrow morreu(x, 79)),$
$F_6.\ \forall x : (\forall t_1 : (\forall t_2 : (( mortal(x) \wedge nasceu(x, t_1) \wedge \mathbf{maq}(-( t_2, t_1), 150)) \rightarrow morto(x,t_2)))),$
$F_7.\ agora = 1998,$
$F_8 \forall x : (\forall t : (morto(x, t) \rightarrow \neg vivo(x, t)),$

$F_9$. $\forall x : (\forall t : (\neg morto(x, t) \rightarrow vivo(x, t))$ ,

$F_{10}$. $\forall x : (\forall t_1 : (\forall t_2 : ((morreu(x, t_1) \wedge \mathbf{maq}(t_2, t_1)) \rightarrow morto(x, t_2))))\}$

$C = \exists t : morreu(Marcos, t)$

**1.**

$P = \{$ $C_1$. $homem(Marcos)$ ,

$C_2$. $Pompeano(Marcos)$ ,

$C_3$. $nasceu(Marcos, 40)$ ,

$C_4$. $\neg homem(x_1) \vee mortal(x_1)$ ,

$C_5$. $\neg Pompeano(x_2)\ morreu(x, 79)$ ,

$C_6$. $entrou\_em\_erupção(vulcão,79)$ ,

$C_7$. $\neg mortal(x) \vee \neg\ nasceu(x, t_1) \vee \neg\ \mathbf{maq}(-(t_2, t_1), 150) \vee morto(x, t_2)$,

$C_8$. $agora = 1998$,

$C_9$. $\neg morto(x, t) \vee \neg vivo(x, t)$ ,

$C_{10}$. $morto(x, t) \vee vivo(x,t)$ ,

$C_{11}$. $\neg morreu(x, t_1) \vee \neg\mathbf{maq}(t_2, t_1)) \vee morto(x, t_2)$

$\}$.

**2.**

$\neg C = \forall t{:}\neg morreu(Marcos, t)$.

Convertendo-se a consulta negada para forma clausal obtemos:

$C_{12}$. $= \neg morreu(Marcos, t)$.

$P = \{$ $C_1$. $homem(Marcos)$ ,

$C_2$. $Pompeano(Marcos)$ ,

$C_3$. $nasceu(Marcos, 40)$ ,

$C_4$. $\neg homem(x_1) \vee mortal(x_1)$ ,

$C_5$. $\neg Pompeano(x_2)\ morreu(x_2, 79)$ ,

$C_6$. $entrou\_em\_erupção(vulcão, 79)$ ,

$C_7$. $\neg mortal(x_3) \vee \neg\ nasceu(x_3, t_1) \vee \neg\ \mathbf{maq}(-(t_2, t_1), 150) \vee morto(x_3, t_2)$,

$C_8$. $agora = 1998$ ,

$C_9$. $\neg morto(x_4, t_3) \vee \neg vivo(x_4, t_3)$ ,

$C_{10}$. $morto(x_5, t_4) \vee vivo(x_5, t_4)$ ,

$C_{11}$. $\neg morreu(x_6, t_5) \vee \neg\mathbf{maq}(t_6, t_5) \vee morto(x_6, t_6)$

$C_{12}$. $\neg morreu(Marcos, t)\ \}$.

**3.**

$C_5$. $\neg morreu(Marcos, t)$

$\{ (Marcos/x_2), (79/t) \}$

$Pompeano(Marcos)$ $C_2$.

**nil**

**4. *Resposta = 79*.**

### Observações

- A resposta pode ser derivada da cadeia de unificações que leva de volta à cláusula inicial.
- A resposta pode ser derivada acrescentando uma fórmula fictícia no processo de resolução. Esta fórmula consiste da consulta negada em forma clausal com um marcador especial. Por exemplo:

$C_5$. $\neg morreu(Marcos,t) \vee \underline{\neg morreu(Marcos, t)}$

$\{ (Marcos/x_2), (79/t) \}$

$Pompeano(Marcos) \vee \underline{\neg morreu(Marcos, 79)}$ $C_2$.

$\underline{\neg morreu(Marcos, 79)}$.

Neste caso, a Resolução pode parar quando só restar a fórmula fictícia.

## Atividades de avaliação

**1.** Considerando o par de Cláusulas-pais abaixo:

$C_1 = \neg P(z_1,a) \vee \neg P(z_1,x) \vee \neg P(x,z_1)$ e

$C_2 = P(z_2,f(z_2)) \vee P(z_2,a)$,

onde o símbolo "a" denota uma constante, encontre uma substituição adequada para os dois conjuntos de fórmulas atômicas componentes (omitindo-se a negação em $C_1$) e, logo após, encontre o resolvente correspondente.

**2.** Em relação ao método de RESOLUÇÃO, explique sucintamente porque é importante dispor de um processo de conversão de uma fórmula qualquer para uma fórmula equivalente expressa em forma normal conjuntiva.

**3.** Porque CASAMENTO é importante no método da RESOLUÇÃO?

**4.** Mostre que a Resolução é consistente; isto é, mostre que o resolvente de duas cláusulas-pais segue logicamente destas cláusulas.

**5.** Considere a seguinte base de conhecimento:

$\forall x : (y : (H(x) \wedge D(y)) \rightarrow F(x, y))$ ,

$\exists y : (G(y) \wedge (\forall z : (R(z) \rightarrow F(y, z))))$ ,

$\forall y : (G(y) \rightarrow D(y))$ ,

$\forall x : (\forall y : (\forall z : ((F(x, y) \wedge F(y, z)) \rightarrow F(x, z))))$.

Use Resolução para responder a seguinte consulta:

$\forall x : (\forall z : ((H(x) \wedge R(z)) \rightarrow F(x, z)))$.

**6.** Usando o primeiro programa em lógica desta Seção, responda à pergunta: *Marcos odiava César?"*

**7.** Anteriormente, mostramos que, dado um conjunto de fatos, havia duas maneiras de provar a declaração $\neg vivo(Marcos,agora)$. Nesta Seção, apresentamos uma prova de Resolução que corresponde a um desses métodos. Use a Resolução para derivar a outra prova da declaração, através da outra cadeia de raciocínio.

**8.** Considere as seguintes sentenças:

"João gosta de todo tipo de comida"

"Maçãs são comida"

"Frango é comida"

"Qualquer coisa que alguém coma e que não cause sua morte é comida"

"Paulo come amendoim e ainda está vivo"

"Susana come tudo o que Paulo come"

a) Traduza essas sentenças em Lógica de Predicados.

b) Converta cada uma das fórmulas acima em forma clausal.

c) Responda a pergunta: *"João gosta de amendoim?"*

d) Use a resolução para responder: *"O que Susana come?"* , isto é, $\exists x : come(Susana, x)$.

**9.** Considere os seguintes fatos:

"Os membros do Clube de Tranca da Rua Elmo são João, Salete, Paulo e Helena"

"João é casado com Salete"

"Paulo é irmão de Helena"

"A esposa ou marido de cada pessoa casada membro do clube tam bém está no Clube"

"A última reunião do Clube foi na casa do João"

a) Represente estes fatos em Lógica de Predicados.

b) A partir dos fatos informados, a maioria das pessoas seria capaz de responder às seguintes perguntas:

*"A última reunião do Clube foi na casa de Salete?"*

*"Helena não é casada?"*

Será que você consegue construir provas de resolução para demonstrar a verdade de cada uma destas declarações, dados os fatos listados anteriormente. Faça-o, se possível. Caso contrário, acrescente os fatos necessários e depois crie as provas

**10.** Assuma os seguintes fatos:

"Carlos gosta de cursos fáceis"

"O curso de ciências é difícil"

"Todos os cursos do departamento de prendas domésticas são fáceis"

"BK 301 é um curso de prendas domésticas"

Use a Resolução para responder à pergunta:

*"De que curso Carlos gostaria?"*

**11.** Nesta Seção respondemos à pergunta: *"Quando Marcos morreu?"*, usando a resolução para mostrar que houve um tempo em que Marcos morreu. Usando os mesmos fatos, e o fato adicional

$$\forall x : (\forall t_1 : (morto(x, t_1) \rightarrow (t_2 : (maq(t_1, t_2) \wedge morreu(x, t_2))))),$$

existe um outro modo de mostrar que houve um tempo em que Marcos morreu.

a) Faça uma prova de resolução desta outra cadeia de raciocínio.

b) Que resposta esta prova dará à pergunta: *"Quando Marcos morreu?"*

**12.** Se um curso é fácil, alguns estudantes no curso são felizes. Se um curso tem exame, nenhum estudante no curso é feliz. Use resolução para mostrar que, se um curso tem exame, o curso não é fácil.

**13.** Qualquer coisa que pode ler é alfabetizada. Alguns golfinhos são inteligentes, mas nenhum golfinho é alfabetizado. Use resolução para mostrar que algumas coisas inteligentes não sabem ler.

**14.** Vitor foi assassinado, e Artur, Bernadete e Karlene são suspeitos. Artur disse que ele não fez isso. Ele disse que Bernadete era amiga da vítima mas que Karlene odiava a vítima. Bernadete disse que ela estava fora da cidade no dia do assassinato e, além disso, ela disse que nem conhecia o rapaz. Karlene disse que ela é inocente e que ela viu Artur e Bernadete com a vítima logo após o assassinato. Assumindo que todo mundo (exceto o assassino) está falando a verdade, use resolução para resolver o crime.

**15.** Sabemos que cavalos são mais rápidos do que cães e que há um cachorrão, chamado Fig, que é mais rápido que todos os coelhos. É conhecido que Centelha é um cavalo e Pernalonga é um coelho. Use resolução para mostrar que Centelha é mais rápido que Pernalonga.

**16.** Considere o seguinte programa:

*"Animal que tem pena não é mamífero"*

*"Animal que têm pêlo é mamífero e não é ave"*

*"Animal que não é mamífero é ave"*

*"Animal que não tem pena têm pêlo"*

*"Sabiá é um animal que voa e não é mamífero"*

*"Pingüim é um animal que é ave e não voa"*

*"Vaca é um animal que é mamífero e não voa"*

*"Morcego é um animal que voa e não é ave"*

Consulte o programa para identificar um animal que voa e não tem pena. Mostre a seqüência de prova usada.

## Sobre os autores

**Gustavo Augusto Lima de Campos:** Cursou graduação em Engenharia Elétrica na Universidade Federal do Pará (1987), mestrado em Engenharia Elétrica na Universidade Federal de Uberlândia (1990) e doutorado em Engenharia Elétrica na Universidade Estadual de Campinas (2003). Atualmente é professor adjunto da Universidade Estadual do Ceará. Tem experiência na área de Ciência da Computação, com ênfase em Inteligência Artificial, atuando principalmente nos seguintes temas: agentes inteligentes, lógica, redes neurais artificiais, sistemas fuzzy, busca e problemas de tomada de decisão.

**Jerffeson Teixeira de Souza:** Recebeu o título de Ph.D. em Ciência da Computação, em 2004, pela School of Information Technology and Engineering (SITE) da University of Ottawa, Canadá. Ele é Bacharel (1998) e Mestre (2000) em Ciência da Computação pela Universidade Federal do Ceará (UFC). Ele é atualmente professor adjunto da Universidade Estadual do Ceará (UECE). Seus interesses de pesquisa são: Otimização em Engenharia de Software, Documentação e Aplicação de Padrões de Software e Estudo de Técnicas e Aplicação de Algoritmos de Mineração de Dados.

Computação

Fiel a sua missão de interiorizar o ensino superior no estado Ceará, a UECE, como uma instituição que participa do Sistema Universidade Aberta do Brasil, vem ampliando a oferta de cursos de graduação e pós-graduação na modalidade de educação a distância, e gerando experiências e possibilidades inovadoras com uso das novas plataformas tecnológicas decorrentes da popularização da internet, funcionamento do cinturão digital e massificação dos computadores pessoais.

Comprometida com a formação de professores em todos os níveis e a qualificação dos servidores públicos para bem servir ao Estado, os cursos da UAB/UECE atendem aos padrões de qualidade estabelecidos pelos normativos legais do Governo Federal e se articulam com as demandas de desenvolvimento das regiões do Ceará.

ISBN 978-85-78264-45-1

UNIVERSIDADE ESTADUAL DO CEARÁ

UAB UNIVERSIDADE ABERTA DO BRASIL

CAPES