Matemática Discreta para Computação e Informática

P. Blauth Menezes

blauth@inf.ufrgs.br

Departamento de Informática Teórica

Instituto de Informática / UFRGS

Matemática Discreta para Computação e Informática

P. Blauth Menezes

2 – Lógica e Técnicas de Demonstração

♦ Lógica Matemática

- básica para qq estudo em Computação e Informática
- em particular, para estudo de Matemática Discreta

♦ Para desenvolver qq algoritmo (qq *software*)

- necessários conhecimentos básicos de Lógica

♦ Existem linguagens de progr. baseadas em Lógica

- desenvolvidas segundo o paradigma lógico
- exemplo: Prolog

- **Diretrizes Curriculares do MEC para Cursos de Computação e Informática**

  Lógica Matemática é uma ferramenta fundamental na definição de conceitos computacionais

- **Para matérias da Área de Formação Tecnológica, como Inteligência Artificial**

  Como base ao estudo da Inteligência Artificial são imprescindíveis conhecimentos de Lógica Matemática, ...

- **Lógica permite definir *Teorema***

- **Por que teoremas e suas demonstrações são fundamentais para a Computação e Informática?**

  - *teorema* (freqüentemente) pode ser visto como
    - problema a ser implementado computacionalmente

  - *demonstração*
    - solução computacional
    - algoritmo o qual *prova-se*, sempre funciona!

- **David Parnas, importante pesquisador internacional e um dos pioneiros da Engenharia de Software**
  - XIII SBES - Seminários Brasileiros de Engenharia de Software

o maior avanço da Engenharia de Software nos últimos dez anos foi os provadores de teoremas

- **Objetivo**
  - introduzir principais conceitos e terminologia necessários para MD
    - *não* é uma abordagem ampla nem detalhada
    - existe uma disciplina específica de Lógica

2 – Noções de Lógica e Técnicas de Demonstração

2.1 Lógica

- 2.1.1 Proposições
- 2.1.2 Conetivos
- 2.1.3 Fórmulas, Ling. Lógica e Tabelas Verdade
- 2.1.4 Lógica nas Linguagens de Programação
- 2.1.5 Tautologia e Contradição
- 2.1.6 Implicação e Equivalência
- 2.1.7 Quantificadores

2.2 Técnicas de Demonstração

- 2.2.1 Prova Direta
- 2.2.1 Prova por Contraposição
- 2.2.1 Prova por Redução ao Absurdo

2.1 Lógica

♦ Estudo centrado em

- Lógica Booleana ou Lógica de Boole
  - ⁎ George Boole: inglês, 1815-1864
  - ⁎ um dos precursores da Lógica

- estudo dos princípios e métodos usados para
  - ⁎ distinguir sentenças verdadeiras de falsas

2 – Noções de Lógica e Técnicas de Demonstração

2.1.1 Proposições

Def: Proposição

Construção (sentença, frase, pensamento) que pode-se atribuir juízo

- tipo de juízo na Lógica Matemática
  - verdadeiro-falso
  - interesse é na “verdade” das proposições

♦ Forma tradicional de tratar com a “verdade”

- dois valores verdade V (verdadeiro) e F (falso)
- proposições só podem assumir esses valores

♦ Denotação do valor verdade de uma proposição p

$$V(p)$$

Exp: Proposição

- Brasil é um país (valor verdade V)
- Buenos Aires é a capital do Brasil (valor verdade F)
- $3 + 4 > 5$ (valor verdade V)
- $7 - 1 = 5$ (valor verdade F)

Ou seja

- V(Brasil é um país) = V
- V(Buenos Aires é a capital do Brasil) = F
- $V(3 + 4 > 5) = V$
- $V(7 - 1 = 5) = F$

Exp: *Não* são proposição

Vá tomar banho.

Que horas são?

Parabéns!

2 – Noções de Lógica e Técnicas de Demonstração

2.1 Lógica

- 2.1.1 Proposições
- 2.1.2 Conetivos
- 2.1.3 Fórmulas, Ling. Lógica e Tabelas Verdade
- 2.1.4 Lógica nas Linguagens de Programação
- 2.1.5 Tautologia e Contradição
- 2.1.6 Implicação e Equivalência
- 2.1.7 Quantificadores

2.2 Técnicas de Demonstração

- 2.2.1 Prova Direta
- 2.2.1 Prova por Contraposição
- 2.2.1 Prova por Redução ao Absurdo

2.1.2 Conetivos

♦ Proposições introduzidas

- proposições atômicas ou átomos
- *não* podem ser decompostas em proposições mais simples

♦ É possível construir proposições mais complexas

- compondo proposições
- usando operadores lógicos ou conetivos

Exp: Proposições Compostas

Windows é sistema operacional **e** Pascal é ling. de programação

Vou comprar um PC **ou** um MAC

Linux **não** é um *software* livre

Se chover canivetes, **então** todos estão aprovados em MD

$A = B$ **se e somente se** $(A \subseteq B$ **e** $B \subseteq A)$

- **Proposições compostas podem ser usadas para construir novas proposições compostas**
  - $A = B$ **se e somente se** $(A \subseteq B$ **e** $B \subseteq A)$

♦ Cinco conetivos que serão estudados

- e (conjunção)
- ou (disjunção)
- não (negação)
- se-então (condicional)
- se-somente-se (bicondicional)

Negação

- **Uma proposição p ou é verdadeira ou é falsa**
- **Negação de uma proposição**
  - introduzindo a palavra *não*
  - prefixando a proposição por *não é fato que* (ou equivalente)

Exp: Negação

Brasil é um país	Brasil *não* é um país
Linux é um *software* livre	Linux *não* é um *software* livre
$3 + 4 > 5$	*Não é fato que* $3 + 4 > 5$

♦ Negação de p

$$\neg p \quad \text{ou} \quad {\sim}p$$

- lê-se: "não p"

♦ Semântica da negação

- se p é verdadeira, então ¬p é falsa
- se p é falsa, então ¬p é verdadeira

♦ Tabela Verdade

- descreve os valores lógicos de uma proposição
- em termos das combinações dos valores lógicos das proposições componentes e dos conetivos usados

Def: Negação

Semântica da Negação ¬p

p	¬p
V	F
F	V

Conjunção

- ♦ **Conjunção de duas proposições p e q**

$$p \land q$$

  - lê-se: "p e q"

- ♦ **Reflete uma noção de simultaneidade**
  - verdadeira, apenas quando p e q são simultaneamente verdadeiras
  - falsa, em qualquer outro caso

Def: Conjunção

Semântica da Conjunção $p \wedge q$

p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

- quatro linhas para expressar todas as combinações de valores lógicos de p e q
- quantas linhas para n proposições?

Exp: Conjunção

Verdadeira

- *Windows* é sist. operacional **e** Pascal é ling. de programação

Falsa

- *Windows* é sistema operacional **e** Pascal é planilha eletrônica
- *Windows* é editor de textos **e** Pascal é ling. de programação
- *Windows* é editor de textos **e** Pascal é planilha eletrônica

Exercício: Conjunção

Suponha que p e q são respectivamente V e F. Valor lógico?

- $p \wedge \neg q$
- $\neg p \wedge q$
- $\neg p \wedge \neg q$

Exercício: Conjunção

Determine o $V(p)$, sabendo que

- $V(q) = V$ e $V(p \land q) = F$

Disjunção

- **Disjunção de duas proposições p e q**

$$p \vee q$$

  - lê-se: "p ou q"

- **Reflete uma noção de "pelo menos uma"**
  - verdadeira, quando pelo menos uma das proposições é verdadeira
  - falsa, somente quando simultaneamente p e q são falsas

Def: Disjunção

Semântica da Disjunção p v q

p	q	p v q
V	V	V
V	F	V
F	V	V
F	F	F

Exp: Disjunção

Verdadeira

- *Windows* é sist. operacional **ou** Pascal é ling. de programação
- *Windows* é sistema operacional **ou** Pascal é planilha eletrônica
- *Windows* é editor de textos **ou** Pascal é ling. de programação

Falsa

- *Windows* é editor de textos **ou** Pascal é planilha eletrônica

Exercício: Disjunção

Suponha que p e q são respectivamente V e F. Valor lógico?

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

Exercício: Disjunção

Determine o $V(p)$, sabendo que

- $V(q) = F$ e $V(p \vee q) = F$

Condição

♦ Condição de duas proposições p e q

$$p \to q$$

- lê-se: "se p então q"

♦ Reflete a noção

- partir de uma premissa p verdadeira
- obrigatoriamente deve-se chegar a uma conclusão q verdadeira
- para que $p \to q$ seja verdadeira

- **Entretanto, partindo de uma premissa falsa**
  - qualquer conclusão pode ser considerada
- **Portanto $p \rightarrow q$ é**
  - falsa, quando p é verdadeira e q é falsa
  - verdadeira, caso contrário

Def: Condição

Semântica da Condição $p \rightarrow q$

p	q	$p \rightarrow q$
V	V	V
V	F	F
F	V	V
F	F	V

Exp: Condição

Verdadeira

- **se** *Windows* é sist. operacional **então** Pascal é ling. de progr.
- **se** *Windows* é editor de textos **então** Pascal é ling. de programação
- **se** *Windows* é editor de textos **então** Pascal é planilha eletrônica

Falsa

- **se** *Windows* é sist. operacional **então** Pascal é planilha eletrônica

Exercício: Condição

Determine o $V(p)$, sabendo que

- $V(q) = F$ e $V(p \rightarrow q) = F$
- $V(q) = F$ e $V(q \rightarrow p) = V$

Exercício: Condição

Determine o $V(p)$ e $V(q)$, sabendo que

- $V(p \rightarrow q) = V$ e $V(p \wedge q) = F$
- $V(p \rightarrow q) = V$ e $V(p \vee q) = F$

Bicondição

♦ **Bicondição de duas proposições p e q**

$$p \leftrightarrow q$$

- lê-se: "p se e somente se q"

♦ **Reflete a noção de condição "nos dois sentidos"**

- considera simultaneamente
  - *ida*: p é premissa e q é conclusão
  - *volta*: q é premissa e p é conclusão

♦ Portanto $p \leftrightarrow q$ é

- verdadeira, quando p e q são ambas verdadeiras ou ambas falsas
- falsa, quando p e q possuem valor verdade distintos

Def: Bicondição

Semântica da Bicondição $p \leftrightarrow q$

p	q	$p \leftrightarrow q$
V	V	V
V	F	F
F	V	F
F	F	V

Exp: Bicondição

Verdadeira

- *Windows* é sist. oper. **se e somente se** Pascal é ling. de progr.
- *Windows* é ed. de textos **se e somente se** Pascal é planilha eletr.

Falsa

- *Windows* é sist. Oper. **se e somente se** Pascal é planilha eletr.
- *Windows* é ed. de textos **se e somente se** Pascal é ling. de progr.

Exercício: Bicondição

Determine o $V(p)$, sabendo que

- $V(q) = V$ e $V(p \leftrightarrow q) = F$
- $V(q) = F$ e $V(q \leftrightarrow p) = V$

Exercício: Bicondição

Determine o $V(p)$ e $V(q)$, sabendo que

- $V(p \leftrightarrow q) = V$ e $V(p \land q) = V$
- $V(p \leftrightarrow q) = V$ e $V(p \lor q) = V$
- $V(p \leftrightarrow q) = F$ e $V(\neg p \lor q) = V$

2 – Noções de Lógica e Técnicas de Demonstração

2.1.3 Fórmulas, Linguagem Lógica e Tabelas Verdade

♦ Fórmulas Lógicas ou simplesmente Fórmulas

- palavras da Linguagem Lógica
- introduzido formalmente adiante
  - quando do estudo da Definição Indutiva
- *informalmente*, sentença lógica corretamente construída sobre o alfabeto cujos símbolos são
  - conetivos ($\wedge$, $\vee$, $\rightarrow$, …)
  - parênteses
  - identificadores (p, q, r, …)
  - constantes, etc.

♦ Se a fórmula contém variáveis

- *não* necessariamente possui valor verdade associado
- valor lógico depende do valor verdade das sentenças que substituem as variáveis na fórmula

Exp: Fórmulas

Suponha p, q e r são sentenças variáveis

- valores verdade constantes V e F
- qualquer proposição
- p, q e r
- $\neg p$, $p \wedge q$, $p \vee q$, $p \rightarrow q$ e $p \leftrightarrow q$
- $p \vee (\neg q)$
- $(p \wedge \neg q) \rightarrow F$
- $\neg(p \wedge q) \leftrightarrow (\neg p \vee \neg q)$
- $p \vee (q \wedge r) \leftrightarrow (p \vee q) \wedge (p \vee r)$

♦ Precedência entre os conetivos

- reduzir os parênteses
- simplificar visualmente

♦ Ordem de precedência entre os conetivos

- entre parênteses, dos mais internos para os mais externos
- negação ($\neg$)
- conjunção ($\wedge$) e disjunção ($\vee$)
- condição ($\rightarrow$)
- bicondição ($\leftrightarrow$)

Exp: Precedência de Conetivos

$p \lor (\neg q)$ $\qquad$ $p \lor \neg q$

$(p \land \neg q) \rightarrow F$ $\qquad$ $p \land \neg q \rightarrow F$

$\neg(p \land q) \leftrightarrow (\neg p \lor \neg q)$ $\qquad$ $\neg(p \land q) \leftrightarrow \neg p \lor \neg q$

$p \lor (q \land r) \leftrightarrow (p \lor q) \land (p \lor r)$

- qualquer omissão de parênteses resulta em ambigüidade
- (por quê?)

♦ Tabelas Verdade

- como construir uma tabela verdade de uma dada fórmula?
- explicitar todas as combinações possíveis dos valores lógicos
  - ∗ das fórmulas atômicas componentes
- fórmula atômica *não*-constante
  - ∗ dois valores lógicos: V e F
- fórmula atômica *constante*
  - ∗ valor verdade fixo (V ou F)

♦ Uma fórmula atômica (não-constante): negação

- tabela: 2 linhas
- 2^1 possíveis combinações dos valores lógicos

♦ Duas fórmulas atômicas (não-constantes): conjunção, condição ...

- tabela: 4 linhas
- 2^2 possíveis combinações dos valores lógicos

♦ n fórmulas atômicas (não-constantes)

- tabela: 2^n linhas
- 2^n possíveis combinações de valores lógicos
- (fácil verificar tal resultado)

Exp: Tabela Verdade

Construção da tabela verdade para a fórmula $p \vee \neg q$

p	q
V	V
V	F
F	V
F	F

p	q	¬q
V	V	F
V	F	V
F	V	F
F	F	V

p	q	¬q	p ∨ ¬q
V	V	F	V
V	F	V	V
F	V	F	F
F	F	V	V

Exp: Tabela Verdade: $p \wedge \neg q \rightarrow F$

Não foi introduzida uma coluna para o valor constante F

- seria redundante (conteria somente F)

p	q	$\neg q$	$p \wedge \neg q$	$p \wedge \neg q \rightarrow F$
V	V	F	F	V
V	F	V	V	F
F	V	F	F	V
F	F	V	F	V

Exp: Tabela Verdade: $p \lor (q \land r) \leftrightarrow (p \lor q) \land (p \lor r)$

p	q	r	$q \land r$	$p \lor (q \land r)$	$p \lor q$	$p \lor r$	$(p \lor q) \land (p \lor r)$	$p \lor (q \land r) \leftrightarrow (p \lor q) \land (p \lor r)$
V	V	V	V	V	V	V	V	V
V	V	F	F	V	V	V	V	V
V	F	V	F	V	V	V	V	V
V	F	F	F	V	V	V	V	V
F	V	V	V	V	V	V	V	V
F	V	F	F	F	V	F	F	V
F	F	V	F	F	F	V	F	V
F	F	F	F	F	F	F	F	V

Exercício: Tabela Verdade

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

2 – Noções de Lógica e Técnicas de Demonstração

2.1 Lógica

2.1.1 Proposições
2.1.2 Conetivos
2.1.3 Fórmulas, Ling. Lógica e Tabelas Verdade
2.1.4 Lógica nas Linguagens de Programação
2.1.5 Tautologia e Contradição
2.1.6 Implicação e Equivalência
2.1.7 Quantificadores

2.2 Técnicas de Demonstração

2.2.1 Prova Direta
2.2.1 Prova por Contraposição
2.2.1 Prova por Redução ao Absurdo

2.1.4 Lógica nas Linguagens de Programação

- **Em geral, LP possuem o tipo de dado lógico (booleano) predefinido**
- **Pascal**
  - tipo de dado é **`boolean`**
  - valores lógicos V e F são **`true`** e **`false`**
  - declaração (definição) das variáveis **`p`**, **`q`** e **`r`**

```
p, q, r: boolean
```

♦ **Já foi introduzido que as noções de**

- igualdade e contido (entre conjuntos)
- pertinência (de um elemento a um conjunto)
- resultam em valores lógicos

♦ **Analogamente, relações entre expressões aritméticas resultam em valores lógicos**

=	(igual)
<	(menor)
<=	(menor ou igual)
>	(maior)
>=	(maior ou igual)

♦ Trechos de programas em Pascal

- (qual o valor lógico resultante?)

```
7 – 1 = 5
```

```
n + 1 > n
```

♦ Conetivos lógicos Pascal (e na maioria das LP)

not	(negação)
and	(conjunção)
or	(disjunção)
<=	(condição)
=	(bicondição)

Exp: Programa em Pascal

Calcular o valor lógico de $p \lor (q \land r)$ para qq valores de p, q e r lidos

```
program valor_logico (input, output);
var p, q, r: boolean;
begin
   read (p, q, r);
   if p or (q and r)
   then write('verdadeiro')
   else write('falso')
end.
```

2 – Noções de Lógica e Técnicas de Demonstração

2.1.5 Tautologia e Contradição

Def: Tautologia, Contradição

Seja w uma fórmula

- Tautologia
  - ∗ w é verdadeira
  - ∗ para qq combinação possível de valores de sentenças variáveis
- Contradição
  - ∗ w é falsa
  - ∗ para qq combinação possível de valores de sentenças variáveis

Exp: Tautologia, Contradição

Suponha p uma fórmula

- p ∨ ¬p é tautologia
- p ∧ ¬p é contradição

p	¬p	p ∨ ¬p	p ∧ ¬p
V	F	V	F
F	V	V	F

2 – Noções de Lógica e Técnicas de Demonstração

2.1.6 Implicação e Equivalência

- **Conetivos condição e bicondição induzem relações entre fórmulas**
  - Condição: implicação
  - Bicondição: equivalência
- **Importância destas relações**
  - Relação de implicação
    - relacionada com o conceito de teorema
  - Relação de equivalência
    - "mesmo significado" entre fórmulas (sintaticamente) diferentes

Def: Relação de Implicação

p e q fórmulas

$$p \Rightarrow q$$

p implica em q

se e somente se

$$p \rightarrow q \text{ é uma tautologia}$$

Exp: Relação de Implicação

(interprete os nomes)

Adição: $p \Rightarrow p \lor q$

Simplificação: $p \land q \Rightarrow p$

p	q	$p \lor q$	$p \rightarrow (p \lor q)$	$p \land q$	$(p \land q) \rightarrow p$
V	V	V	V	V	V
V	F	V	V	F	V
F	V	V	V	F	V
F	F	F	V	F	V

Def: Relação de Equivalência

p e q fórmulas

$$p \Leftrightarrow q$$

p é equivalente a q

se e somente se

$p \leftrightarrow q$ é uma tautologia

Exp: Relação de Equivalência

p	q	r	$q \wedge r$	$p \vee (q \wedge r)$	$p \vee q$	$p \vee r$	$(p \vee q) \wedge (p \vee r)$	$p \vee (q \wedge r) \leftrightarrow (p \vee q) \wedge (p \vee r)$
V	V	V	V	V	V	V	V	V
V	V	F	F	V	V	V	V	V
V	F	V	F	V	V	V	V	V
V	F	F	F	V	V	V	V	V
F	V	V	V	V	V	V	V	V
F	V	F	F	F	V	F	F	V
F	F	V	F	F	F	V	F	V
F	F	F	F	F	F	F	F	V

Exp: …Relação de Equivalência

Distributividade do conetivo ou sobre o conetivo

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

Exercício

- verificar se o conetivo e distribui-se sobre o conetivo ou

♦ Exemplos de equivalência que seguem

- importantes para o estudo das Técnicas de Demonstração

Exp: Bicondição × Condição $\quad p \leftrightarrow q \Leftrightarrow (p \rightarrow q) \land (q \rightarrow p)$

Bicondição pode ser expressa por *duas* condições: ida e volta

p	q	$p \leftrightarrow q$	$p \rightarrow q$	$q \rightarrow p$	$(p \rightarrow q) \land (q \rightarrow p)$	$(p \leftrightarrow q) \leftrightarrow (p \rightarrow q) \land (q \rightarrow p)$
V	V	V	V	V	V	V
V	F	F	F	V	F	V
F	V	F	V	F	F	V
F	F	V	V	V	V	V

Exp: Contraposição $p \rightarrow q \Leftrightarrow \neg q \rightarrow \neg p$

p	q	$p \rightarrow q$	$\neg p$	$\neg q$	$\neg q \rightarrow \neg p$	$p \rightarrow q \leftrightarrow \neg q \rightarrow \neg p$
V	V	V	F	F	V	V
V	F	F	F	V	F	V
F	V	V	V	F	V	V
F	F	V	V	V	V	V

Exp: Redução ao Absurdo $\quad p \rightarrow q \Leftrightarrow p \land \neg q \rightarrow F$

p	q	$p \rightarrow q$	$\neg q$	$p \land \neg q$	$p \land \neg q \rightarrow F$	$p \rightarrow q \leftrightarrow p \land \neg q \rightarrow F$
V	V	V	F	F	V	V
V	F	F	V	V	F	V
F	V	V	F	F	V	V
F	F	V	V	F	V	V

♦ Exercícios CIC

- Idempotência
  - $p \land p \Leftrightarrow p$
  - $p \lor p \Leftrightarrow p$
- Comutativa
  - $p \land q \Leftrightarrow q \land p$
  - $p \lor q \Leftrightarrow q \lor p$
- Associativa
  - $p \land (q \land r) \Leftrightarrow (p \land q) \land r$
  - $p \lor (q \lor r) \Leftrightarrow (p \lor q) \lor r$
- Distributiva
  - $p \lor (q \land r) \Leftrightarrow (p \lor q) \land (p \lor r)$

♦ ...Exercícios CIC

- Dupla negação
  - $\neg\neg p \Leftrightarrow p$

- DeMorgan
  - $\neg(p \land q) \Leftrightarrow \neg p \lor \neg q$
  - $\neg(p \lor q) \Leftrightarrow \neg p \land \neg q$

- Absorção
  - $p \land (p \lor q) \Leftrightarrow p$
  - $p \lor (p \land q) \Leftrightarrow p$

♦ Exercícios ECP

- Bastam os conetivos ¬ e ∧
  - Prove que os conetivos estudados pode ser expresso usando somente ¬ e ∧
- Conetivos EXORe NAND
  - Prove que tais conetivos podem ser expressos usando os conetivos já estudados

x	y	x EXOR y	x NAND y
V	V	F	F
V	F	V	V
F	V	V	V
F	F	F	V

2 – Noções de Lógica e Técnicas de Demonstração

2.1.7 Quantificadores

♦ **Proposição sobre um conjunto de valores**

$$n > 1$$

- dependendo do valor de n
- assume valor verdadeiro ou falso
  - para cada valor de n considerado, é uma proposição diferente

♦ **Quantificadores**

- dada uma proposição sobre um conjunto de valores
- freqüentemente é desejável quantificar os valores a serem considerados

Proposição Sobre um Conjunto

Def: Proposição sobre um Conjunto

Proposição Sobre A

- valor lógico depende do elemento $x \in A$ considerado

Exp: Proposição sobre $\mathbf{N}$

$n > 1$

$n! < 10$

$n + 1 > n$

$2n$ é ímpar

Quais proposições são verdadeiras para qualquer $n \in \mathbf{N}$?

$p(x)$

- proposição p a qual descreve alguma propriedade de $x \in A$

♦ Toda a proposição p sobre A determina 2 conjuntos

- Conjunto verdade de p

$$\{ x \in A \mid p(x) \text{ é verdadeira} \}$$

- Conjunto falsidade de p

$$\{ x \in A \mid p(x) \text{ é falsa} \}$$

Exp: Conjuntos Verdade e Falsidade — Suponha $\mathbf{N}$

$n > 1$

- $\{ 2, 3, 4,\dots \}$ conjunto verdade
- $\{ 0, 1 \}$ conjunto falsidade

$n! < 10$

- $\{ 0, 1, 2, 3 \}$ conjunto verdade
- $\{ n \in \mathbf{N} \mid n > 3 \}$ conjunto falsidade

$n + 1 > n$

- $\mathbf{N}$ conjunto verdade (o próprio conjunto universo)
- $\varnothing$ conjunto falsidade

"$2n$ é ímpar"

- $\varnothing$ conjunto verdade
- $\mathbf{N}$ conjunto falsidade (o próprio conjunto universo)

♦ Uma proposição p sobre A é

- Tautologia
  - ∗ se p(x) é verdadeira para qualquer $x \in A$
  - ∗ conjunto verdade é A
- Contradição
  - ∗ se p(x) é falsa para qualquer $x \in A$
  - ∗ conjunto falsidade é A

Exp: Tautologia, Contradição Conjunto universo $\mathbf{N}$

$n! < 10$

- *não* é tautologia nem contradição
  - para $n = 0$, a fórmula é verdadeira
  - para $n = 4$, a fórmula é falsa

$n + 1 > n$

- é tautologia
- conjunto verdade é o conjunto universo $\mathbf{N}$

"$2n$ é ímpar"

- é contradição
- conjunto falsidade é o conjunto universo $\mathbf{N}$

Quantificador

- **Com freqüência, para uma proposição p(x)**
  - desejável quantificar os valores de x que devem ser considerados
- **Quantificadores são usados em Lógica (suponha**
  - Quantificador universal, simbolizado por $\forall$

$$(\forall x \in A)(p(x)) \qquad (\forall x \in A)\, p(x) \qquad \forall x \in A,\, p(x)$$

  - Quantificador existencial, simbolizado por $\exists$

$$(\exists x \in A)(p(x)) \qquad (\exists x \in A)\, p(x) \qquad \exists x \in A,\, p(x)$$

♦ Denotação alternativa para $(\forall x \in A)\ p(x)$ e $(\exists x \in A)\ p(x)$

- quando é claro o conjunto de valores

$$(\forall x)(p(x)) \qquad (\forall x)\ p(x) \qquad \forall x,\ p(x)$$

$$(\exists x)(p(x)) \qquad (\exists x)\ p(x) \qquad \exists x,\ p(x)$$

♦ Leitura de $(\forall x \in A)\ p(x)$

qualquer x, p(x) ou *"para todo x, p(x)*

♦ Leitura de $(\exists x \in A)\ p(x)$

existe pelo menos um x tal que p(x) ou *existe x tal que p(x)*

♦ Como a leitura induz, o valor verdade de uma proposição quantificada é

- $(\forall x \in A)\ p(x)$ é verdadeira
  - ⁎ se $p(x)$ for verdadeira para *todos* os elementos de A
- $(\exists x \in A)\ p(x)$ é verdadeira
  - ⁎ se $p(x)$ for verdadeira para *pelo menos um* elemento de A

Def: Quantificador Universal, Quantificador Existencial

Seja $p(x)$ proposição lógica sobre um conjunto A

Quantificador Universal: $(\forall x \in A)\ p(x)$ é

- verdadeira, se o conjunto verdade for A
- falsa, caso contrário

Quantificador Existencial: $(\exists x)\ p(x)$é

- verdadeira, se o conjunto verdade for *não*-vazio
- falsa, caso contrário

Exp: Quantificador Universal, Quantificador Existencial

$(\forall n \in \mathbf{N})(n < 1)$ é falsa

$(\exists n \in \mathbf{N})(n < 1)$ é verdadeira

$(\forall n \in \mathbf{N})(n! < 10)$ é falsa

$(\exists n \in \mathbf{N})(n! < 10)$ é verdadeira

$(\forall n \in \mathbf{N})(n + 1 > n)$ é verdadeira

$(\exists n \in \mathbf{N})(n + 1 > n)$ é verdadeira

$(\forall n \in \mathbf{N})(2n$ é par$)$ é verdadeira

$(\exists n \in \mathbf{N})(2n$ é par$)$ é verdadeira

Sempre que uma proposição quantificada universalmente é verdadeira

- a mesma proposição quantificada existencialmente é verdadeira
- vale sempre ???

♦ Generalização de p(x)

$$p(x_1, x_2, \ldots, x_n)$$

- p descreve alguma propriedade de $x_1 \in A_1$, $x_2 \in A_2, \ldots, x_n \in A_n$
- cada elemento $x_1, x_2, \ldots, x_n$ pode ser individualmente quantificado
  - a ordem dos quantificadores existencial e universal
  - pode alterar o valor verdade da proposição
- Exemplo, para o conjunto universo $\mathbf{N}$, tem-se que
  - $(\forall n)(\exists m)(n < m)$ é verdadeira
  - $(\exists m)(\forall n)(n < m)$ é falsa

Obs: Existe pelo menos um × Existe um único

É comum quantificar existencialmente de forma *única*

- simbolizado por $\exists!$
- existe um elemento e este é *único*
  - *não* pode existir mais de um

$(\exists! n \in \mathbf{N})(n < 1)$ é verdadeira

$(\exists n \in \mathbf{N})(n < 1)$ é verdadeira

$(\exists! n \in \mathbf{N})(n! < 10)$ é falsa

$(\exists n \in \mathbf{N})(n! < 10)$ é verdadeira

$(\exists! n \in \mathbf{N})(n + 1 > n)$ é falsa

$(\exists n \in \mathbf{N})(n + 1 > n)$ é verdadeira

$(\exists! n \in \mathbf{N})(2n$ é par$)$ é falsa

$(\exists n \in \mathbf{N})(2n$ é par$)$ é verdadeira

Obs: ...Existe pelo menos um $\times$ Existe um único

$\exists!$ é equivalentemente a

$$(\exists! x)\, p(x) \Leftrightarrow (\exists x)\, p(x) \wedge (\forall x)(\forall y)(\, (p(x) \wedge p(y) \rightarrow x = y)\,)$$

- primeiro termo: *existe*
- segundo termo: *único*

Negação de Proposições Quantificadas

Negação de proposição quantificada é intuitiva

$$(\forall x \in A)\ p(x)$$

$(\forall x \in A)\ p(x)$ é V, se $p(x)$ for V para todos os elementos de A

Negação: *não* é V para *todos* os elemento de A

- *existe* pelo menos um x tal que não é fato que p(x)

$$\sim((\forall x \in A)\ p(x)) \Leftrightarrow (\exists x)\ \sim p(x)$$

Raciocínio análogo para $(\exists x \in A)\ p(x)$

$$\sim((\exists x \in A)\ p(x)) \Leftrightarrow (\forall x)\ \sim p(x)$$

Exp: Negação de Proposições Quantificadas

$\sim((\forall n \in \mathbf{N})(n < 1)) \Leftrightarrow (\exists n \in \mathbf{N})(n \geq 1) \Leftrightarrow V$

$\sim((\exists n \in \mathbf{N})(n < 1)) \Leftrightarrow (\forall n \in \mathbf{N})(n \geq 1) \Leftrightarrow F$

$\sim((\forall n \in \mathbf{N})(n! < 10)) \Leftrightarrow (\exists n \in \mathbf{N})(n! \geq 10) \Leftrightarrow V$

$\sim((\exists n \in \mathbf{N})(n! < 10)) \Leftrightarrow (\forall n \in \mathbf{N})(n! \geq 10) \Leftrightarrow F$

$\sim((\forall n \in \mathbf{N})(n + 1 > n)) \Leftrightarrow (\exists n \in \mathbf{N})(n + 1 \leq n) \Leftrightarrow F$

$\sim((\exists n \in \mathbf{N})(n + 1 > n)) \Leftrightarrow (\forall n \in \mathbf{N})(n + 1 \leq n) \Leftrightarrow F$

$\sim((\forall n \in \mathbf{N})(2n$ é par$)) \Leftrightarrow (\exists n \in \mathbf{N})(2n$ não é par$) \Leftrightarrow F$

$\sim((\exists n \in \mathbf{N})(2n$ é par$)) \Leftrightarrow (\forall n \in \mathbf{N})(2n$ não é par$) \Leftrightarrow F$

- **Negação pode ser estendida para proposições que dependem de n elementos individualmente quantificados**

Exp: Negação de Proposições Quantificadas

Proposições quantificadas

- $(\forall n)(\exists m)(n < m)$ é verdadeira
- $(\exists m)(\forall n)(n < m)$ é falsa

Negação

- $(\exists n)(\forall m)(n \geq m)$ é falsa
- $(\forall m)(\exists n)(n \geq m)$ é verdadeira

2 – Noções de Lógica e Técnicas de Demonstração

2.1 Lógica

- 2.1.1 Proposições
- 2.1.2 Conetivos
- 2.1.3 Fórmulas, Ling. Lógica e Tabelas Verdade
- 2.1.4 Lógica nas Linguagens de Programação
- 2.1.5 Tautologia e Contradição
- 2.1.6 Implicação e Equivalência
- 2.1.7 Quantificadores

2.2 Técnicas de Demonstração

- 2.2.1 Prova Direta
- 2.2.1 Prova por Contraposição
- 2.2.1 Prova por Redução ao Absurdo

2.2 Técnicas de Demonstração

♦ **Teorema: uma proposição do tipo**

$$p \rightarrow q$$

- prova-se ser verdadeira sempre (tautologia)

$$p \Rightarrow q$$

  - $*$ p - hipótese
  - $*$ q - tese

♦ Corolário

- teorema que é conseqüência quase direta de outro já demonstrado

♦ Lema

- teorema auxiliar
- resultado importante para a prova de outro

♦ Teoremas são fundamentais em Computação e Informática

- Exemplo: permite verificar se uma implementação é correta
- um algoritmo que prova-se, sempre funciona

♦ Fundamental identificar claramente a hipótese e a tese

- exemplo

0 é o único elemento neutro da adição em $\mathbf{N}$

- reescrita identificando claramente a hipótese e a tese

se 0 é elemento neutro da adição em $\mathbf{N}$*,*
então 0 é o único elemento neutro da adição em $\mathbf{N}$

♦ Na demonstração de que $p \Rightarrow q$

- hipótese p é *suposta* verdadeira
  - ∗ *não* deve ser demonstrada

- **Todas as teorias possuem um conjunto de premissas (hipóteses)**
  - são *supostas* verdadeiras
  - sobre as quais todo o raciocínio é construído
- **Teoria dos Conjuntos**
  - baseada em uma premissa: noção de elemento é suposta
  - algumas abordagens consideram a noção de conjunto como sendo uma premissa
- **Hipótese de Church**
  - Computação e Informática é construída sobre tal premissa

Obs: Hipótese de Church × Computação e Informática

Algoritmo, procedimento efetivo ou função computável

- um dos conceitos mais fundamentais da Computação e Informática
- intuitivamente

uma seqüência finita de instruções, as quais podem ser realizadas mecanicamente, em um tempo finito

Tal intuição *não* corresponde a um conceito formal de algoritmo

Início do século XX

- pesquisadores se dedicaram a formalizar tal conceito
- diversas formalizações matemáticas foram desenvolvidas
  - ∗ 1936, Alan Turing propôs o modelo Máquina de Turing

Obs: …Hipótese de Church × Computação e Informática

1936: Alonzo Church apresentou a Hipótese de Church

qualquer função computável pode ser processada por uma Máquina de Turing, ou seja, existe um procedimento expresso na forma de uma Máquina de Turing capaz de processar a função

Como a noção intuitiva de algoritmo *não* é matematicamente precisa

- impossível demonstrar formalmente se a Máquina de Turing é o mais genérico dispositivo de computação
- entretanto, foi mostrado que todos os demais modelos possuem, no máximo, a mesma capacidade computacional

Obs: …Hipótese de Church × Computação e Informática

Para todos o desenvolvimento subseqüentes

- Hipótese de Church é *suposta*
- *premissa básica* para toda a Computação e Informática

Se for encontrado um modelo mais geral do que a Máquina de Turing??

- pela semântica do $\rightarrow$
- estudos desenvolvidos continuam válidos (por quê?)

Teoria da Computação estuda

- Máquina de Turing, Hipótese de Church e conceitos correlatos

- **Um teorema pode ser apresentado na forma $p \leftrightarrow q$**
  - uma técnica usual é provar em separado
    - ida $p \rightarrow q$
    - volta $q \rightarrow p$

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

- **Para um teorema $p \rightarrow q$ existem diversas técnicas para provar (demonstrar) que, de fato, $p \Rightarrow q$**
  - Prova Direta
  - Prova por Contraposição
  - Prova por Redução ao Absurdo ou Prova por Absurdo
  - Prova por Indução

- **Prova por indução**
  - aplicação particular do Princípio da Indução Matemática
  - capítulo específico adiante
- **Demais tipos de prova são introduzidos a seguir**
- **Ao longo da disciplina**
  - cada demonstração é um exemplo das técnicas
  - cada exercícios de demonstração é um exercício das técnicas

♦ Para qualquer técnica de demonstração

- especial atenção aos quantificadores
- provar a proposição

$$(\forall x \in A)\ p(x)$$

  - ⁎ provar para *todo* $x \in A$
  - ⁎ mostrar para um elemento $a \in A$ é um exemplo e *não* uma prova

- provar a proposição

$$(\exists x \in A)\ p(x)$$

  - ⁎ basta provar para *um* $a \in A$
  - ⁎ *um* exemplo *é* uma prova (compare com o caso universal)

2 – Noções de Lógica e Técnicas de Demonstração

2.2.1 Prova Direta

Def: Prova Direta ou Demonstração Direta

Pressupõe verdadeira a hipótese

- a partir desta, prova ser verdadeira a tese

Exp: Prova Direta

a soma de dois números pares é um número par

Reescrevendo na forma de $p \rightarrow q$

se n e m são dois números pares quaisquer,
então $n + m$ é um número par

Qualquer par n pode ser definido como $n = 2r$, para algum natural r

Suponha que n e m são dois pares *quaisquer*

Então existem $r, s \in \mathbf{N}$ tais que

$$n = 2r \quad \text{e} \quad m = 2s$$

Portanto

$$n + m = 2r + 2s = 2(r + s)$$

Como a soma de dois naturais $r + s$ é natural

$$n + m = 2(r + s)$$

Logo, $n + m$ *é* um número par

2 – Noções de Lógica e Técnicas de Demonstração

2.1 Lógica

2.1.1 Proposições
2.1.2 Conetivos
2.1.3 Fórmulas, Ling. Lógica e Tabelas Verdade
2.1.4 Lógica nas Linguagens de Programação
2.1.5 Tautologia e Contradição
2.1.6 Implicação e Equivalência
2.1.7 Quantificadores

2.2 Técnicas de Demonstração

2.2.1 Prova Direta
2.2.1 Prova por Contraposição
2.2.1 Prova por Redução ao Absurdo

2.2.2 Prova por Contraposição

♦ **Baseia-se no resultado denominado *contraposição***

$$p \rightarrow q \Leftrightarrow \neg q \rightarrow \neg p$$

Def: Prova (ou Demonstração) por Contraposição

Para provar $p \rightarrow q$, prova-se

$\neg q \rightarrow \neg p$ (prova direta)

- a partir de $\neg q$
- obter $\neg p$

Exp: Prova por Contraposição

$$n! > (n+1) \ \rightarrow \ n > 2$$

Por contraposição

$$n \leq 2 \ \rightarrow \ n! \leq n+1$$

Muito *simples*!!!

- testar para os casos $n = 0$, $n = 1$ e $n = 2$
- exercício

2 – Noções de Lógica e Técnicas de Demonstração

2.2.3 Prova por Redução ao Absurdo

♦ **Baseia-se no resultado *redução ao absurdo***

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

Def: Prova (Demonstração) por Redução ao Absurdo

Ou simplesmente Prova (Demonstração) por Absurdo

Para provar $p \rightarrow q$, prova-se

$$(p \wedge \neg q) \rightarrow F \qquad \text{(prova direta)}$$

- supor a hipótese p
- supor a *negação* da tese $\neg q$
- concluir uma *contradição* (em geral, $q \wedge \neg q$)

♦ Prova por contra-exemplo

- é demonstração por absurdo
  - ∗ construção da contradição $q \wedge \neg q$
  - ∗ em geral, apresentação de um contra-exemplo

Exp: Prova por Redução ao Absurdo

0 é o único elemento neutro da adição em $\mathbf{N}$

Reescrevendo na forma de $p \rightarrow q$:

se 0 é elemento neutro da adição em $\mathbf{N}$,
então 0 é o único elemento neutro da adição em $\mathbf{N}$

Uma prova por redução ao absurdo

Exp: ...Prova por Redução ao Absurdo

- suponha
  - (hipótese) 0 é o elemento neutro da adição em $\mathbf{N}$
  - (negação da tese) 0 *não* é o único neutro da adição em $\mathbf{N}$
- seja e um *outro* neutro da adição em $\mathbf{N}$ tal que $e \neq 0$
- como 0 é elemento neutro, para qq $n \in \mathbf{N}$
  - $n = 0 + n = n + 0$
  - em particular, para $n = e$: $e = 0 + e = e + 0$
- como e é elemento neutro, para qq $n \in \mathbf{N}$
  - $n = n + e = e + n$
  - em particular, para $n = 0$: $0 = 0 + e = e + 0$

Exp: …Prova por Redução ao Absurdo

- portanto, como $e = 0 + e = e + 0$ e $0 = 0 + e = e + 0$
  - ∗ pela transitividade da igualdade: $e = 0$
  - ∗ *contradição!!!* pois foi suposto que $e \neq 0$

Logo, é absurdo supor que o neutro da adição em $\mathbf{N}$ não é único

Portanto, 0 é o único neutro da adição em $\mathbf{N}$

Matemática Discreta para Computação e Informática

P. Blauth Menezes

Matemática Discreta para Computação e Informática

P. Blauth Menezes

blauth@inf.ufrgs.br

Departamento de Informática Teórica

Instituto de Informática / UFRGS

Informática Teórica

UFRGS