Programação Lógica

Lógica Proposicional Booleana

Prof. Edson Alves

Campus UnB Gama: Faculdade de Ciências e Tecnologias em Engenharia

George Boole

 * 1815 † 1864

George Boole

 * 1815 † 1864

The Mathematic Analysis of Logic (1847)

George Boole

 * 1815 † 1864

The Mathematic Analysis of Logic (1847)

⋆ Proposta de formalização da lógica por meio da matemática

George Boole

 * 1815 † 1864

The Mathematic Analysis of Logic (1847)

⋆ Proposta de formalização da lógica por meio da matemática

⋆ O livro introduz os fundamentos da lógica proposicional booleana

George Boole

* 1815 † 1864

The Mathematic Analysis of Logic (1847)

⋆ Proposta de formalização da lógica por meio da matemática

⋆ O livro introduz os fundamentos da lógica proposicional booleana

⋆ Ele resgata e expande estes fundamentos no seu livro mais conhecido, *An Investigation of the Laws of Thought (1849)*

Lógica Proposicional Booleana

Lógica Proposicional Booleana

Termos primitivos

Lógica Proposicional Booleana

Termos primitivos

- Proposição

Lógica Proposicional Booleana

Termos primitivos

- Proposição
- Verdadeiro

Lógica Proposicional Booleana

Termos primitivos

- Proposição
- Verdadeiro
- Falso

Lógica Proposicional Booleana

Termos primitivos

- ⋆ Proposição
- ⋆ Verdadeiro
- ⋆ Falso

Axiomas

Lógica Proposicional Booleana

Termos primitivos

- Proposição
- Verdadeiro
- Falso

Axiomas

- Princípio do terceiro excluído

Lógica Proposicional Booleana

Termos primitivos

- ⋆ Proposição
- ⋆ Verdadeiro
- ⋆ Falso

Axiomas

- ⋆ Princípio do terceiro excluído
- ⋆ Princípio da não-contradição

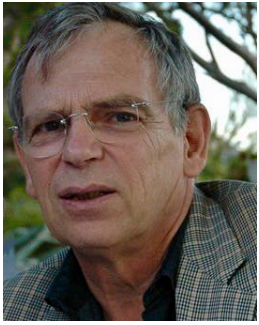
Prolog (1972)

Prolog (1972)

Proponentes

Prolog (1972)

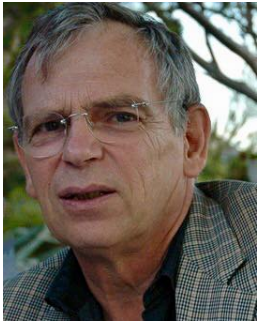
Proponentes



Alain Colmerauer

Prolog (1972)

Proponentes

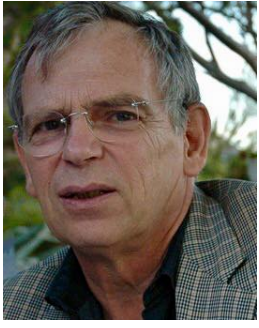


Alain Colmerauer

Philippe Roussel

Prolog (1972)

Proponentes



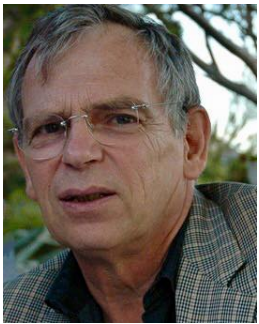
Alain Colmerauer

Philippe Roussel

Inspiração

Prolog (1972)

Proponentes



Alain Colmerauer

Philippe Roussel

Inspiração

Robert Kowalski

SWI Prolog

SWI Prolog

- Prolog é uma contração da expressão “PROgramming in LOGic”

SWI Prolog

- Prolog é uma contração da expressão “PROgramming in LOGic”
- Tem raízes na lógica de primeira ordem

SWI Prolog

- Prolog é uma contração da expressão “PROgramming in LOGic”
- Tem raízes na lógica de primeira ordem
- O SWI-Prolog pode ser instalado por meio do comando

```
$ sudo apt-get install swi-prolog
```

SWI Prolog

- Prolog é uma contração da expressão “PROgramming in LOGic”
- Tem raízes na lógica de primeira ordem
- O SWI-Prolog pode ser instalado por meio do comando

```
$ sudo apt-get install swi-prolog
```

- O interpretador (*listener*) Prolog pode ser invocado com o comando

```
$ prolog
```

Valores lógicos em Prolog

Valores lógicos em Prolog

- Prolog implementa os termos primitivos *verdadeiro* e *falso* por meio dos predicados `true/0` e `false/0`

Valores lógicos em Prolog

⋆ Prolog implementa os termos primitivos *verdadeiro* e *falso* por meio dos predicados `true/0` e `false/0`

```
?- true.
true.

?- false.
false.
```

Valores lógicos em Prolog

- Prolog implementa os termos primitivos *verdadeiro* e *falso* por meio dos predicados `true/0` e `false/0`

```
?- true.
true.

?- false.
false.
```

- Prolog faz distinção entre maiúsculas e minúsculas

Valores lógicos em Prolog

- Prolog implementa os termos primitivos *verdadeiro* e *falso* por meio dos predicados `true/0` e `false/0`

```
?- true.
true.

?- false.
false.
```

- Prolog faz distinção entre maiúsculas e minúsculas

```
?- True.
% ... 1,000,000 ..... 10,000,000 years later
%
%       >> 42 << (last release gives the question)
```

Conectivos da lógica proposicional booleana

Conectivos da lógica proposicional booleana

Operação	Leitura	Definição

Conectivos da lógica proposicional booleana

Operação	Leitura	Definição
$\neg a$	não a	Inverte o valor lógico de a

Conectivos da lógica proposicional booleana

Operação	Leitura	Definição
$\neg a$	não a	Inverte o valor lógico de a
$a \vee b$	a ou b	Falso apenas se a e b são ambos falsos

Conectivos da lógica proposicional booleana

Operação	Leitura	Definição
$\neg a$	não a	Inverte o valor lógico de a
$a \vee b$	a ou b	Falso apenas se a e b são ambos falsos
$a \wedge b$	a e b	Verdadeiro apenas se a e b são ambos verdadeiros

Conectivos da lógica proposicional booleana

Operação	Leitura	Definição
$\neg a$	não a	Inverte o valor lógico de a
$a \vee b$	a ou b	Falso apenas se a e b são ambos falsos
$a \wedge b$	a e b	Verdadeiro apenas se a e b são ambos verdadeiros
$a \rightarrow b$	se a, então b	Falso apenas se a é verdadeiro e b é falso

Conectivos da lógica proposicional booleana

Operação	Leitura	Definição
$\neg a$	não a	Inverte o valor lógico de a
$a \vee b$	a ou b	Falso apenas se a e b são ambos falsos
$a \wedge b$	a e b	Verdadeiro apenas se a e b são ambos verdadeiros
$a \rightarrow b$	se a, então b	Falso apenas se a é verdadeiro e b é falso
$a \leftrightarrow b$	a se, e somente se, b	Verdadeiro se ambos tem mesmo valor lógico

Fatos

Fatos

- Fatos são os predicados mais simples da linguagem Prolog

Fatos

- Fatos são os predicados mais simples da linguagem Prolog
- Eles correspondem a proposições verdadeiras

Fatos

- Fatos são os predicados mais simples da linguagem Prolog
- Eles correspondem a proposições verdadeiras
- A sintaxe para a declaração do fato `pred/N` é

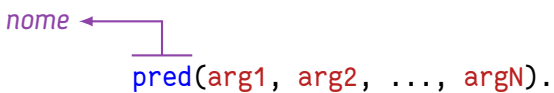
Fatos

- Fatos são os predicados mais simples da linguagem Prolog
- Eles correspondem a proposições verdadeiras
- A sintaxe para a declaração do fato `pred/N` é

```
pred(arg1, arg2, ..., argN).
```

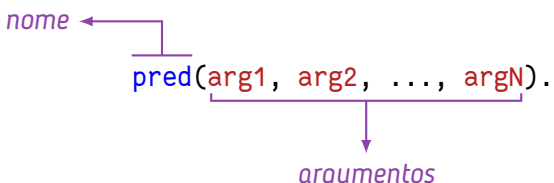
Fatos

- Fatos são os predicados mais simples da linguagem Prolog
- Eles correspondem a proposições verdadeiras
- A sintaxe para a declaração do fato `pred/N` é



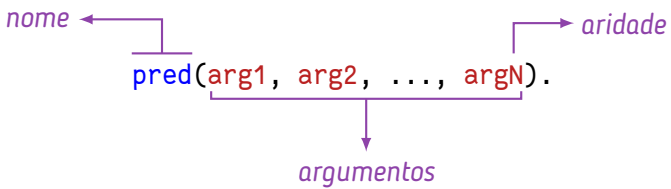
Fatos

- Fatos são os predicados mais simples da linguagem Prolog
- Eles correspondem a proposições verdadeiras
- A sintaxe para a declaração do fato `pred/N` é



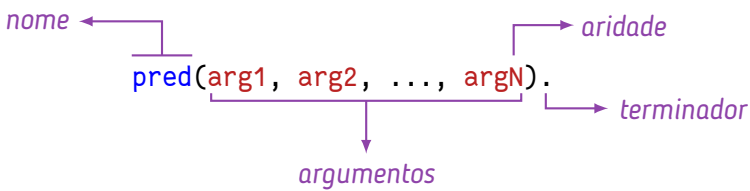
Fatos

- Fatos são os predicados mais simples da linguagem Prolog
- Eles correspondem a proposições verdadeiras
- A sintaxe para a declaração do fato `pred/N` é



Fatos

- Fatos são os predicados mais simples da linguagem Prolog
- Eles correspondem a proposições verdadeiras
- A sintaxe para a declaração do fato `pred/N` é



Declarando fatos em Prolog

Declarando fatos em Prolog

- Em Prolog, os fatos devem ser declarados em arquivos, que serão lidos posteriormente pelo interpretador

Declarando fatos em Prolog

- Em Prolog, os fatos devem ser declarados em arquivos, que serão lidos posteriormente pelo interpretador

- A extensão deste arquivos deve ser '`.pl`'

Declarando fatos em Prolog

⋆ Em Prolog, os fatos devem ser declarados em arquivos, que serão lidos posteriormente pelo interpretador

⋆ A extensão deste arquivos deve ser '`.pl`'

⋆ O interpretador pode ler um arquivo em sua inicialização, por meio da opção '`-s`':

```
$ prolog -s source.pl
```

Declarando fatos em Prolog

⋆ Em Prolog, os fatos devem ser declarados em arquivos, que serão lidos posteriormente pelo interpretador

⋆ A extensão deste arquivos deve ser '`.pl`'

⋆ O interpretador pode ler um arquivo em sua inicialização, por meio da opção '`-s`':

```
$ prolog -s source.pl
```

⋆ Os predicados `consult/1` e `reconsult/1` podem ser usados para carregar ou recarregar um arquivo em uma sessão interativa do interpretador Prolog

```
% Implementação do conectivos lógicos em Prolog
% Proposições não declaradas são consideradas falsas

and(true, true).

or(true, true).
or(true, false).
or(false, true).

not(false).
```

Programas em Prolog

Programas em Prolog

- Em Prolog, os programas correspondem a consultas na base de fatos carregada no interpretador

Programas em Prolog

- Em Prolog, os programas correspondem a consultas na base de fatos carregada no interpretador

- Cada consulta é feita diretamente no interpretador e deve indicar o nome do predicado, seus os argumentos e o terminador (ponto final)

Programas em Prolog

⋆ Em Prolog, os programas correspondem a consultas na base de fatos carregada no interpretador

⋆ Cada consulta é feita diretamente no interpretador e deve indicar o nome do predicado, seus os argumentos e o terminador (ponto final)

⋆ Se a consulta consiste em um único fato, o interpretador retornará verdadeiro se o fato em questão faz parte da base de fatos, ou falso, caso contrário

Programas em Prolog

⋆ Em Prolog, os programas correspondem a consultas na base de fatos carregada no interpretador

⋆ Cada consulta é feita diretamente no interpretador e deve indicar o nome do predicado, seus os argumentos e o terminador (ponto final)

⋆ Se a consulta consiste em um único fato, o interpretador retornará verdadeiro se o fato em questão faz parte da base de fatos, ou falso, caso contrário

⋆ A opção '`-s`' e os predicados `consult/1` e `reconsult/1` manipulam a base de fatos do interpretador, adicionando novos fatos ou atualizando os fatos existentes

```
% Exemplo de sessão interativa Prolog
% O interpretador é iniciado com o comando
%
%     $ prolog
%

?- consult('codes/conectivos.pl').
true.

?- and(true, false).
false.

?- or(false, true).
true.

?- not(true).
false.
```

Proposições compostas

Proposições compostas

- Embora a implementação dos conectivos lógicos por meio de fatos esteja conceitualmente correta, ela não permite a expressão de proposições compostas

Proposições compostas

⋆ Embora a implementação dos conectivos lógicos por meio de fatos esteja conceitualmente correta, ela não permite a expressão de proposições compostas

⋆ Por exemplo, a proposição composta $(F \vee V) \wedge V$ é verdadeira, porém em Prolog temos

```
?- and(or(true, false), true).
false.
```

Proposições compostas

⋆ Embora a implementação dos conectivos lógicos por meio de fatos esteja conceitualmente correta, ela não permite a expressão de proposições compostas

⋆ Por exemplo, a proposição composta $(F \vee V) \wedge V$ é verdadeira, porém em Prolog temos

```
?- and(or(true, false), true).
false.
```

⋆ Isso ocorre devido ao comportamento da unificação em Prolog

Proposições compostas

⋆ Embora a implementação dos conectivos lógicos por meio de fatos esteja conceitualmente correta, ela não permite a expressão de proposições compostas

⋆ Por exemplo, a proposição composta $(F \vee V) \wedge V$ é verdadeira, porém em Prolog temos

```
?- and(or(true, false), true).
false.
```

⋆ Isso ocorre devido ao comportamento da unificação em Prolog

⋆ A linguagem oferece, porém, suporte a conjunções, disjunções e negações

Predicados de controle

Predicados de controle

- São predicados que implementam estruturas de controle e, em geral, são traduzidos pelo compilador

Predicados de controle

- São predicados que implementam estruturas de controle e, em geral, são traduzidos pelo compilador

- O predicado `,/2` corresponde à conjunção

Predicados de controle

- São predicados que implementam estruturas de controle e, em geral, são traduzidos pelo compilador

- O predicado `,/2` corresponde à conjunção

- O predicado `;/2` corresponde à disjunção

Predicados de controle

⋆ São predicados que implementam estruturas de controle e, em geral, são traduzidos pelo compilador

⋆ O predicado `,/2` corresponde à conjunção

⋆ O predicado `;/2` corresponde à disjunção

⋆ O predicado `\+/1` corresponde à negação e é verdadeiro apenas quando não há evidência ou prova para seu argumento

Predicados de controle

- São predicados que implementam estruturas de controle e, em geral, são traduzidos pelo compilador

- O predicado `,/2` corresponde à conjunção

- O predicado `;/2` corresponde à disjunção

- O predicado `\+/1` corresponde à negação e é verdadeiro apenas quando não há evidência ou prova para seu argumento

- Os predicados `true/0` e `false/0` também são predicados de controle

Proposições compostas em Prolog

Proposições compostas em Prolog

⋆ Retomando o exemplo anterior, a proposição composta $(F \vee V) \wedge V$ pode ser expressa em Prolog por meio dos predicados de controle da seguinte forma:

```
?- (false ; true), true.
true.
```

Proposições compostas em Prolog

⋆ Retomando o exemplo anterior, a proposição composta $(F \vee V) \wedge V$ pode ser expressa em Prolog por meio dos predicados de controle da seguinte forma:

```
?- (false ; true), true.
true.
```

⋆ Os predicados de controle são implementados como operadores, de modo que podem ser usados com notação infixada

Proposições compostas em Prolog

⋆ Retomando o exemplo anterior, a proposição composta $(F \vee V) \wedge V$ pode ser expressa em Prolog por meio dos predicados de controle da seguinte forma:

```
?- (false ; true), true.
true.
```

⋆ Os predicados de controle são implementados como operadores, de modo que podem ser usados com notação infixada

⋆ Embora não seja obrigatório, é fortemente recomendado o uso de parêntesis em expressões que envolvem o predicado `;/2`

Referências

- **dtonhofer/prolog_notes.** *Negação as Failure,* acesso em 18/02/2026.
- **MERRIT**, Dennis. *Adventure in Prolog, Amzi!,* 191 pgs, 2017.
- **SWI-Prolog.** *https://www.swi-prolog.org/,* acesso em 10/02/2026.
- **WOLFRAM**, Stephen. *George Boole: A 200-Year View,* acesso em 10/02/2026.