1. Introdução

Introduzida entre os anos de 1971 e 1973, Prolog é uma linguagem de programação de forte importância na ciência da computação por ter sido a primeira linguagem lógica aritmética, com aplicação em diversas áreas, tendo como principais destaques a inteligência artificial, seu uso na interpretação da linguística natural humana e na comprovação automática de teoremas.

Entre suas principais características está o paradigma lógico/descritivo, o qual é a maior referência, e sua forte base matemática. Prolog é uma linguagem bem diferente do habitual - que se caracteriza por estipular maneiras de chegar à solução de um problema fazendo passo a passo - pois se limita às definições pré-estabelecidas pelo programador durante a criação do programa.

Nesse trabalho iremos apresentar características e fatos históricos da linguagem, bem como suas semelhanças e principais diferenças com as outras linguagens de programação. Além disso, nos aprofundaremos na questão do paradigma de Programação em Lógica Matemática e exemplificaremos seu uso na programação.

2. Histórico

Prolog (Programmation em Logique, em francês) foi introduzido entre 1971 e 1973 por Alain Colmerauer e Phillipe Roussel da Universidade de Aix-Marseille, em Provença, França, e recebeu a colaboração de Robert Kowalski da Universidade de Edinburgh, Escócia. Com o propósito inicial de traduzir linguagens naturais, a linguagem permitiu fórmulas para ser interpretada de tal forma que uma conclusão lógica poderia ser alcançada, atraindo a atenção da sociedade de computação por ser a primeira linguagem de programação lógica aritmética.

A ideia nasceu a partir da pesquisa da época na área de inteligência artificial de conseguir que computadores comprovem teoremas automaticamente. Em 1970, Colmerauer, que era Professor Assistente de ciência de Computação na Universidade de Montreal, convidou Roussel e Kowalski para trabalhar em seu projeto de tradução automática TAUM (Traduction Automatique de l'Université de Montréal). O projeto acabou evoluindo e durou cerca de 3 anos até chegar a ser publicado, inicialmente continha 610 cláusulas e fazia

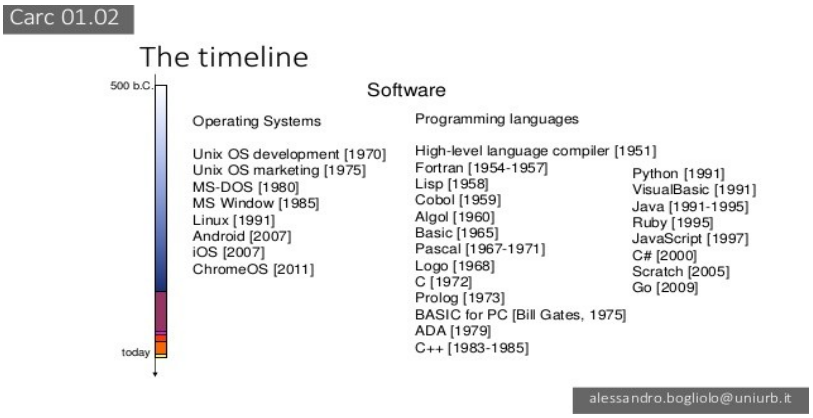
inferências por meio de pronomes, artigos e preposições, sendo então a primeira máquina capaz de realizar uma comunicação com humanos baseado somente na “lógica”.

Em 1977, David Warren, especialista em Inteligência Artificial da Universidade de Edimburgh, implementou uma versão eficiente de Prolog, batizada de Prolog-10, e tornou-se, a partir de então, uma escolha natural para a resolução de problemas que envolvem a representação simbólica de objetos e relações entre objetos. O “Warren Abstract Machine” (WAM) criado por Warren é até hoje o método padrão de implementação da linguagem.

Em 1975 a equipe recebeu uma tarefa: cada um dos indivíduos deveria traduzir duas páginas de Fortran para a linguagem de máquina e fazer compilar na máquina T1600. A equipe conseguiu realizar a tarefa e assim terminou a história do nascimento de Prolog. A linguagem só receberia selo de alta qualidade após 1981, quando cresceu rapidamente na comunidade de informática devido ao lançamento do “Projeto Quinta Geração” no Japão. O "Japanese Institute for New Generation Computing Technology” queria providenciar uma tecnologia complementar de hardware na forma de máquinas de interferência lógica (Projeto Quinta Geração) e selecionou programação lógica como a linguagem de software de habilitação.

Atualmente, o Prolog é utilizado em diversas aplicações na área de computação simbólica, incluindo-se aí: bases de dados relacionais, sistemas especialistas, lógica matemática, prova automática de teoremas, resolução de problemas abstratos e geração de planos, processamento de linguagem natural, projeto de arquiteturas, logística, resolução de equações simbólicas, construção de compiladores, análise bioquímica e projeto de fármacos.

Foto1- Prolog no Contexto da Computação



Fonte: (CArcMOOC...2016)

3. Paradigma

Os paradigmas de programação expressam a estruturação e execução dos programas de uma determinada linguagem de programação classificando-as de acordo com suas funcionalidades. A linguagem Prolog pertence ao paradigma de Programação em Lógica Matemática, esta lógica envolve a unificação da expressividade dos sistemas formais e a dedutividade de sistemas de prova matemática formal.

A lógica é uma ferramenta mais confiável para responder questões. Um problema pode ser interpretado como uma teoria e é justamente a lógica que permite descobrir se tal questão é uma nova teoria ou consequente. A primeira proposta do uso de programação com lógica matemática foi de John McCarthy e a primeira linguagem derivada desta inovação foi a Planner, baseada em backtracking, definida então como a precursora do Prolog.

A criação da Planner abriu a questão sobre o rumo que a programação lógica tomaria. Isso porque John McCarthy propunha que a epistemologia de sistemas de computadores, que estuda os princípios, hipóteses e resultados desses sistemas, fosse fundamentada na lógica matemática, enquanto os procedimentos processuais já eram abordados pelo MIT sob a liderança de Marvin Minsky e Seymour Papert.

A ideia de que a computação é uma dedução controlada. (KOWALSKI, 1988). É um slogan atribuído à Pat Hayes, juntamente com o contraponto da tese desenvolvida por Carl Hewitt de que a dedução lógica seria incapaz de executar computação concorrente em sistemas abertos, dá mais um passo no impasse entre as abordagens lógica e procedimental. A solução dessa questão, por fim, é que cada abordagem possui uma semântica diferente, assim, definimos este paradigma como de Programação em Lógica Matemática, tendo semântica própria com base na Teoria dos Modelos.

4. Características principais

Prolog não é uma linguagem orientada a objetos ou procedimental como muitas popularmente utilizadas hoje, ela é declarativa. Uma linguagem declarativa pode ser descrita como uma linguagem cujo funcionamento baseia-se na predeterminação de fatos e regras. Fatos são um conjunto de dados em essência, já as regras definem relações lógicas que combinam os fatos de modo a compor uma estrutura para resolver um problema e substituir as estruturas de controle.

Os predicados também compõem a linguagem, já que ela se baseia em um subconjunto de cálculo de predicados de primeira ordem (CPPO). Um predicado é composto por uma "cabeça", caracterizada por uma sentença, e um conjunto de argumentos que serão analisados. Os predicados compõem os fatos e regras da linguagem.

Outra característica interessante do Prolog é a possibilidade de se utilizar o modo interativo, onde se consulta os dados baseando-se nas regras propostas como consultas a uma base de dados normal. Dessa forma é possível testar a estrutura corrente e estudá-la com maior praticidade.

Os tipos de dados são outro ponto característico da linguagem, isso porque todos eles são compactados em apenas um: o termo. Para se diferenciar o que seriam de fato tipos, considera-se a forma como o termo é declarado. Os identificadores trabalham, obviamente, com um escopo definido, onde variáveis podem assumir valores diferentes durante a execução e qualquer outro objeto deve manter seu valor inicial.

5. Relacionamento com outras linguagens

Prolog, por se tratar de uma linguagem de paradigma diferente da maioria das linguagens de programação mais populares, se difere bastante dessas em quase tudo, como em seu escopo, tipos de dados e operações.

5.1 *Linguagens semelhantes*

Devido ao seu paradigma (lógico) ser bem menos utilizado que outros, como o procedural e a orientação à objetos, linguagens semelhantes são bem raras, sendo a grande maioria alguma variação de Prolog. Entre eles se destacam linguagens como Datalog e Visual Prolog.

5.1.1 *Datalog*

Inicialmente introduzida em 1977 por Hervé Gallaire e Jack Minker, Datalog é sintaticamente um subconjunto de Prolog. É principalmente utilizada em querys para consultar bases de dados dedutíveis. Entre principais divergências estão limitações no uso da negação e da recursão, e o fato de Datalog não ser uma linguagem Turing-completa (não pode ser usada para resolver todo e qualquer problema computável).

5.1.2 *Visual Prolog*

Também conhecido como *Turbo Prolog,* é uma linguagem baseada em Prolog que, em contraste, é fortemente tipada e apresenta múltiplos paradigmas. Entre os principais objetivos dessa linguagem está a simplificação da programação em paradigma lógico, através da aproximação da linguagem de paradigmas mais populares, eliminando algumas dificuldades da programação nesse paradigma.

5.2 *Linguagens contraditórias*

Uma linguagem que está em direto confronto com Prolog no ramo de inteligência artificial é a linguagem Lisp.

Lisp é uma família de linguagens de programação de computadores. E os mais conhecidos dialetos de Lisp usados para programação de uso geral hoje são Common Lisp e Scheme. O nome LISP vem de “LISt Processing” e, como sugere, a principal estrutura de dados de Lisp é a lista encadeada. Na verdade, toda a fonte é escrita usando listas (usando notação de prefixo) ou, mais corretamente, listas entre parênteses (chamadas de expressões-s).

Apesar das duas linguagens serem as mais usadas em IA, elas possuem sérias diferenças. O Lisp é uma linguagem funcional, enquanto o Prolog é uma linguagem lógica de programação e declarativa. O Lisp é muito flexível devido à sua rápida prototipação e recursos de macro, permitindo, assim, estender a linguagem para se adequar ao problema em questão. Nas áreas de IA, gráficos e interfaces de usuário, o Lisp tem sido usado extensivamente por causa dessa capacidade de prototipagem rápida. No entanto, devido às suas capacidades de programação lógica (já devidamente construída na linguagem), o Prolog é ideal para problemas de inteligência artificial com raciocínio simbólico, banco de dados e aplicativos de análise de idioma. A escolha de um sobre o outro depende completamente do tipo de problema de IA que precisa ser resolvido.

6. Exemplos

6.1 Hello World

Foto 2- Hello World em Prolog

```
?- write('Hello World!'), nl.
Hello World!
true.

?-
```

Fonte: Wikipédia

6.2 Quicksort

Foto 3- Quicksort em Prolog

```
split(H, [A|X], [A|Y], Z) :-
  order(A, H), split(H, X, Y, Z).
split(H, [A|X], Y, [A|Z]) :-
  not(order(A, H)), split(H, X, Y, Z).
split(_, [], [], []).
quicksort([], X, X).
quicksort([H|T], S, X) :-
  split(H, T, A, B),
  quicksort(A, S, [H|Y]),
  quicksort(B, Y, X).
```

Fonte: Wikipédia

6.3 Torre de Hanoi

Foto 4- Torre de Hanoi em Prolog

```
hanoi(N) :- move(N, left, center, right).
move(0, _, _, _) :- !.
move(N, A, B, C) :-
  M is N-1,
  move(M, A, C, B), inform(A, B), move(M, C, B, A).
inform(X, Y) :-
  write('move a disc from the '),write(X), write(' pole to the '), write(Y), write(' pole'),
  nl.
```

Fonte: Wikipédia

6.4 Exemplo de cláusula simples

Foto 5- Exemplo cláusula simples

```
1. Here are some simple clauses.

likes(mary,food).
likes(mary,wine).
likes(john,wine).
likes(john,mary).

The following queries yield the specified answers.

 | ?- likes(mary,food).
yes.
 | ?- likes(john,wine).
yes.
 | ?- likes(john,food).
no.
```

Fonte: Computer Science University of Toronto

6.5 Exemplo de unificação

```
% The = sign in Prolog represents unification, so:

?- 2 = 3.                          % False - equality test
?- X = 3.                          % X = 3 - assignment
?- X = 2, X = Y.                   % X = Y = 2 - two assignments
                                   % Note Y is assigned to, even though it is
                                   % on the right hand side, because it is free
?- X = 3, X = 2.                   % False
                                   % First acts as assignment and binds X=3
                                   % Second acts as equality because X is bound
                                   % Since 3 does not equal 2, gives False
                                   % Thus in Prolog variables are immutable
?- X = 3+2.                        % X = 3+2 - unification can't do arithmetic
?- X is 3+2.                       % X = 5 - "is" does arithmetic.
?- 5 = X+2.                        % This is why = can't do arithmetic -
                                   % because Prolog can't solve equations
?- 5 is X+2.                       % Error. Unlike =, the right hand side of IS
                                   % must always be bound, thus guaranteeing
                                   % no attempt to solve an equation.
?- X = Y, X = 2, Z is Y + 3.       % X = Y, Y = 2, Z = 5.
                                   % X = Y are both free, so Prolog remembers
                                   % it. Therefore assigning X will also
                                   % assign Y.
```

Fonte: Learn X in Y Minutes (Website)

7. Considerações Finais

Nesse trabalho queríamos demonstrar de forma sucinta a linguagem de programação Prolog. No fim, conseguimos sintetizar a linguagem demonstrando suas principais características, histórico, paradigmas e linguagens que tem algum relacionamento com ela.

Ao longo do trabalho descobrimos a funcionalidade e especificidade de Prolog, seus empecilhos e a dificuldade de programação. Aprendemos sobre lógica aritmética, paradigma de lógica, cláusulas de Horn, relações e outros. Descobrimos que Prolog tem suas vantagens mas também precisa de características consideradas “impuras” como o cut e input/output. Conseguimos inferir nosso aprendizado até agora de amarrações e escopo e aprendemos o máximo possível sobre “backtracking”.

Por fim, esperamos que esse trabalho seja esclarecedor sobre os principais pontos da linguagem Prolog, a ponto de sabermos o suficiente caso precisemos da linguagem algum dia e que desperte ao leitor curiosidade para pesquisar mais ao fundo.

8. Bibliografia

WATT, David A. **Programming Language Design Concepts.** Chichester: John Wiley & Sons Ltd., 2004

PEREIRA, Fernando C.N.; SHIEBER, Stuart M. **Prolog and Natural Language Analysis.** Massachussets: Microtone Publishing, 2005. Disponível em: <http://www.mtome.com/Publications/PNLA/prolog-digital.pdf>. Acesso em: 02 abr. 2019

ENDRISS, Ulle. **An Introduction to Prolog Programming**. 2018. Disponível em: <https://staff.science.uva.nl/u.endriss/teaching/prolog/prolog.pdf > Acesso em: 02 abr. 2019

9. Referências

BARANAUSKAS, José Augusto. **Sintaxe e Semântica de Programas Prolog.** Disponível em: <http://dcm.ffclrp.usp.br/~augusto/teaching/ia/IA-Prolog-Sintaxe-Semantica.pdf> Acesso em: 02 abr. 2019

Prolog. ln: Wikipédia: a enciclopédia livre. Disponível em: <https://pt.wikipedia.org/wiki/Prolog> . Acesso em: 02 abr. 2019

Datalog. ln: Wikipédia: a enciclopédia livre. Disponível em: <https://pt.wikipedia.org/wiki/Prolog> . Acesso em: 02 abr. 2019

COLMERAUER, A.; ROUSSEL, P. **The birth of Prolog.** Disponivel em: <http://alain.colmerauer.free.fr/alcol/ArchivesPublications/PrologHistory/19november92.pdf>. Acesso em: 02 abr. 2019

KOWALSKI, Robert A. **The Early Years of Logic Programming**. Disponivel em: <http://www.doc.ic.ac.uk/~rak/papers/the%20early%20years.pdf> . Acesso em: 02 abr. 2019

ALBUQUERQUE, Eduardo S.de. **Prolog.** Disponível em: <http://www.inf.ufg.br/~eduardo/lp/alunos/prolog/prolog.html>. Acesso em: 02 abr.2019

A HISTÓRIA DO PROLOG PROGRAMMING LANGUAGE. Disponível em: <http://ptcomputador.com/P/computer-programming-languages/88505.html>. Acesso em: 02 abr. 2019

ROSEBRUGH, Robert. **A Brief History of Prolog.** Disponível em: <https://www.mta.ca/~rrosebru/oldcourse/371199/prolog/history.html>. Acesso em: 02 abr. 2019

PROLOG. 2019. Disponível em: <https://www.cleverism.com/skills-and-tools/prolog/> Acesso em: 02 abr. 2019

ROUCHY, Philippe. **Aspects of Prolog history: Logic Programming and Professional Dynamics.** Disponível em:< https://www.researchgate.net/publication/277325585_Aspects_of_Prolog_history_Logic_Programming_and_Professional_Dynamics>. Acesso em: 02 abr. 2019

INDIKA. **Difference Between Prolog and Lisp.** 20 maio 2011. Disponível em:<https://www.differencebetween.com/difference-between-prolog-and-lisp/> Acesso em: 02 abr. 2019

CArcMOOC 01.02 - Brief history of computing. [S.1.]: Computer Architecture,2016. Disponível em: <https://www.slideshare.net/alessandrobogliolo/carcmooc-0102-brief-history-of-computing> Acesso em: 01 abr. 2019

SOME SIMPLE PROLOG EXAMPLES. Disponível em: <http://www.cs.toronto.edu/~sheila/384/w11/simple-prolog-examples.html>. Acesso em: 02 abr. 2019

LEARN PROLOG IN Y MINUTES. Disponível em: <https://learnxinyminutes.com/docs/prolog/>. Acesso em: 02 abr. 2019

XAVIER, Gley F. C. In: XAVIER, Gley Fabiano Cardoso. **Lógica de programação**. São Paulo: Senac, 2007. p. 25

WATT, David A. Logic programming. In: WATT, David A. **Programming Language Design Concepts.** Chichester: John Wiley & Sons Ltd., 2004. Cap.15, p.393-411