PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e de Informática

Trabalho de Linguaguem de Programaçao - Linguagem Prolog*

Arthur Ladislau Pereira[1]
Ernesto Athayde de Queiroz[2]
Ricardo Xavier Sena[3]

*Artigo apresentado ao Instituto de Ciências Exatas e Informática da Pontifícia Universidade Católica de Minas Gerais como pré-requisito para obtenção do título de Bacharel em Ciência da Computação.

[1]Aluno, Ciência da Computação, Brasil, arthur.ladislau@sga.pucminas.br.
[2]Aluno, Ciência da Computação, Brasil, ernesto.queiroz@sga.pucminas.br.
[3]Aluno, Ciência da Computação, Brasil, ricardo.sena@sga.pucminas.br.

Sumário

Lista de Figuras

1 INTRODUÇÃO

Prolog é uma linguagem de programação que pertence aos paradigmas lógico e declarativo. O nome Prolog foi dado por Phillipe Roussel em 1972 sendo um acrônimo de *Programmation en Logique*.

Figura 1 – Logo Prolog

É uma linguagem de uso geral que consiste numa linguagem puramente lógica, o *Prolog puro*, e numa linguagem concreta que acrescenta componentes extra-lógicos ao Prolog puro. É principalmente utilizada para implementação de interpretadores de Linguagem natural e em Inteligência Artificial pela fácil declaração de relações e implementação não muito verbosa.

Hoje se usa principalmente a implementação SWI-Prolog, da universidade de Amsterdã, para propósitos acadêmicos por ser rica em características (bibliotecas, GUI, *Multithreading*, etc.) e ferramentas (incluindo uma IDE). SWI-Prolog vem sendo desenvolvida desde 1987 e funciona nas plataformas UNIX, Windows e Macintosh.

Figura 2 – Logo SWI-Prolog

2 HISTÓRICO

O projeto Prolog foi iniciado em 1970 por Alain Colmerauer e Philippe Roussel, inspirado pelo artigo *"A machine-oriented logic based on the resolution principle"* de Alan Robinson baseando-se na interpretação processual de Robert Kowalski sobre Cláusulas de Horn com o propósito inicial de processar linguagens naturais, mais especificamente francês. Uma Versão preeliminar da linguagem foi desenvolvida ao final de 1971.

Em 1972, Roussel implementou uma resolução SLD de Kowalski, sendo a mais interessante de todas as implementações prévias. Era particularmente boa para processos não determinísticos pois seu *modus operandi* em pilha era mais semelhante ao de uma linguagem de programação convencional. Nesse período Kowalski colaborou diretamente com a equipe, que estava tendo problemas com a cláusula de Horn. Após essa visita Roussel implementou o primeiro sistema Prolog em Algol-W enquanto Colmerauer desenvolveu o primeiro grande programa da linguagem o "sistema de comunicação homem-máquina". Aqui temos as declarações e as perguntas inseridas no sistema seguidas das respostas dadas pelo sistema:

```
Todo psiquiatra é uma pessoa.
Todas as pessoas que ele analisa, estão doentes.
Jacques é um psiquiatra em Marseille.
Jacques é uma pessoa?
Onde está Jacques?
Jacques está doente?
Sim.
Em Marseille.
Eu não sei.
```

Ou no original em francês:

```
TOUT PSYCHIATRE EST UNE PERSONNE.
CHAQUE PERSONNE QU'IL ANALYSE, EST MALADE.
JACQUES EST UN PSYCHIATRE A *MARSEILLE.
EST-CE QUE *JACQUES EST UNE PERSONNE?
OU EST *JACQUES?
EST-CE QUE *JACQUES EST MALADE?
OUI.
A MARSEILLE.
JE NE SAIS PAS.
```

Finalmente em 1973, após a publicação do sistema de comunicação homem-máquina e desenvolvimento substancial na primeira versão de Prolog, Roussel e Colmerauer inicaram o desenvolvimento de uma nova versão, mais orientada a uma linguagem de programação e não só um sistema de dedução automatizado. Com mais ajuda de Kowalski e mais colaboradores da

Universidade de Edimburgo, foi implementado a versão definitiva de Prolog com a maioria das funcionalidades hoje presentes em Prolog puro.

Entre os anos 1974 e 1975 Prolog foi largamente copiada e difundida, pelas várias pessoas que tomavam interesse e levavam cópias seja diretamente de Marselha ou de outros intermediários como Edimburgo, além da distribuição feita pelos autores. Segundo David Warren, Prolog não foi de fato distríbuida mas na verdade, "escapou"e se "multiplicou".

2.1 Autores

2.1.1 Alain Colmerauer

Alain Colmerauer (1941-2017) foi um cientista da computação francês graduado no Instituto Nacional Politécnico de Grenoble, ganhou um PhD do Ensimag (*École nationale supérieure d'informatique et de mathématiques appliquées de Grenoble*) em Grenoble e foi professor assistente na Universidade de Montreal no inicio de sua carreira e se tornou professor na universidade de Maselha onde desenvolveu Prolog.

Figura 3 – Alain Colmerauer

2.1.2 Philippe Roussel

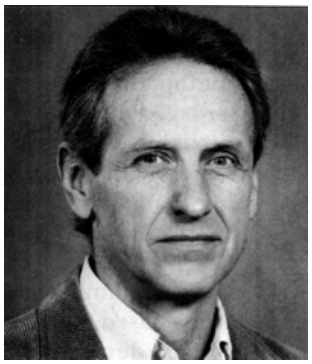
Philippe Roussel (1945) é um doutor em Ciência da Computação considerado um dos fundadores da programação lógica, fez parte da equipe de desenvolvimento de Prolog em Maselha. Roussel foi quem batizou a linguagem.

2.1.3 Robert Kowalski

Robert Kowalski (1941) é um mestre em matemática pela universidade de Stanford e doutor em ciência da computação pela universidade de Edimburgo, onde trabalhou como pesquisador durante todo o desenvolvimento de Prolog e mudou para o *Imperial College London*

desde 1975 como cientista da computação.

Figura 4 – Robert Kowalski



2.2 Influência

Prolog é influenciada por uma linguagem chamada Planner. Alguns conceitos que foram emprestados:

- ***Backward-chaining***: um método de inferência em que se parte do objetivo e se resolve "empilhando"os literais e resolvendo a pilha até ter uma resposta válida.
- ***Negação por falha***: um método de inferência com o que consiste em derivar $\neg P$ da falha de verificação de P.
- ***General Backtracking***: um método de execução em que se constrói uma árvore de candidatos para a solução de um objetivo e os abandona "*backtracks*"assim que a solução se torna inviável.
- ***Usar diferentes nomes para referenciar diferentes entidades***.

3 PARADIGMAS

Prolog incorpora o paradigma declarativo, mais especificamente o paradigma lógico, em seu conceito. Programação declarativa refere-se ao estilo de construir programas usando lógica sem descrever o fluxo e, principalmente, sem utilizar comandos, o paradigma declarativo é antagônico ao paradigma imperativo.

3.1 Paradigma Lógico

O sentido da programação lógica é trazer o estilo da lógica matemática à programação de computadores. Vários problemas são naturalmente expressos como problemas lógicos, e a notação nos proporciona uma maneira de demonstrar se uma questão é verdadeira ou falsa.

4 CARACTERÍSTICAS

4.1 Cláusulas de Horn

Cláusula de Horn é um tipo de cláusula lógica (disjunção de Literais) com no máximo um literal (fórmula atómica, ou atomo, ou a negação do atomo) positivo. Qualquer cláusula de Horn pertence a uma de quatro categorias:

- ***Regra***: 1 literal positivo, pelo menos um literal negativo. Uma regra possui a forma $\neg P_1 \vee \neg P_2 \vee ... \vee \neg P_k \vee Q$ é logicamente equivalente a $[P_1 \wedge P_2 \wedge ... \wedge p_k] \implies Q$.
- ***Fato*** ou ***Unidade***: 1 literal positivo e 0 negativos.
- ***Objetivo Negado***: nenhum literal positivo e ao menos um negativo. Em quase todas as implementações da cláusulas Horn, o objetivo negado nada mais é do que a negação da declaração fornecida.
- ***Cláusula Nula***: 0 literais positivos e 0 negativos, aparece somente ao fim da prova de uma resolução.

Prolog puro é uma implementação do algoritmo com as seguintes especificações:

- As cláusulas em γ são ordenadas pelo programador, e a busca é realizada na ordem *depth-first*.
- Os Literais negativos em cada cláusula são ordenados pelo programador. Quando um objetivo negativo P é resolvido com a cláusula C , os literais negativos em C são colocados em ordem *no inicio* de P.
- A operação "*pick*"recupera o primeiro literal em P. Assim, P é implementado como uma Pilha *Last In First Out*.

Para contextualizar as especificações acima considere que um programa em Prolog é uma base de conhecimento γ.

4.2 Sintaxe

A sintaxe do prolog é simples. Esta é construída a partir de formulações com os predicados lógicos.
1º Tipo: São as “Questões”, isto é, uma pergunta à uma base de conhecimento (A ideia de um teorema a ser mostrado)

```
?-  >(3,2). /* ou ?- 3 > 2. Ou seja, 3 é maior que 2 */
?- Yes
```

2º Tipo: São os fatos (Algo sempre verdadeiro, encontrado na base de conhecimento)

```
?- listing(homem).
homem(joao).
homem(jose).
homem(jedro).
Yes
```

3º Tipo: São as regras (Verdades ou teoremas condicionais)

```
listing(mortal).
mortal(A) :-  homem(A).
/* Para demonstrar que um algum A é mortal,
preciso demonstrar que A é homem*/
Yes
```

Resumindo: Um programa feito em prolog é constituído de fatos, questões e regras.

4.3 Tipos de Dados

Prolog não emprega tipos de dados do mesmo modo que as linguagens de programação mais comuns normalmente fazem. Todos os dados são tratados como sendo de um único tipo, Termo, cuja natureza depende da forma como esse termo foi declarado. Ou seja, os elementos léxicos utilizados na sua declaração determinam se esse termo será um número, um texto, uma variável, uma estrutura complexa e assim por diante.

4.3.1 Átomos

As constantes de texto são introduzidas por meio de átomos. Um átomo é uma sequência constituída de letras, números e underscore, mas iniciando com uma letra minúscula. Se um átomo não alfanumérico é necessário, pode-se usar qualquer sequência entre aspas simples (ex: 'um átomo contendo espaços'). Os átomos podem ser definidos das seguintes maneiras:

Começando com letra minúscula:

```
pedro   henrique_iv
```

Como uma sequência de caracteres entre aspas simples:

```
'quem é você?'  'eu não sei.'
```

4.3.2 Números

Um número é uma sequência de dígitos, permitindo também os sinais de . (para números reais), - (número negativo) e e(notação científica). Algumas implementações do Prolog não fazem distinção entre inteiros e números reais. Exemplos:

```
321   3.21
```

4.3.3 Variáveis

Variáveis são declaradas da mesma forma que átomos, porém iniciando com uma letra maiúscula ou underscore.No ambiente Prolog uma variável não é um contêiner cujo valor pode ser atribuído (como ocorre nas linguagens imperativas). Seu comportamento é mais próximo de um padrão, que é incrementalmente especificado pela unificação. Exemplos:

```
X    Nome    Rei_da_Espanha
```

4.3.4 Termos Compostos

Termos compostos são a única forma de se expressar estruturas de dados complexas em Prolog. Um termo composto consiste de uma cabeça, também chamada funtor (que é obrigatoriamente um átomo) e parâmetros (de quaisquer tipos) listados entre parênteses e separados por vírgulas.

O número de parâmetros, chamado aridade do termo, é significativo. Um termo é identificado por sua cabeça e aridade, normalmente escrita como funtor/aridade. Exemplos:

```
arthur
```

é um átomo, e pode ser considerado um funtor de aridade 0.

```
bonito(arthur)
```

é um termo composto por um funtor de aridade 1 (bonito), com um parâmetro.

```
pai(arthur, ricardo)
```

é um termo composto por um funtor de aridade 1 (pai), com dois parâmetros (arthur e ricardo).

4.3.5 Listas

Uma lista não é um tipo de dados à parte, mas sim definida por uma construção recursiva (usando o termo '.'/2):

- o átomo [] é uma lista vazia.
- se T é uma lista e H é um elemento, então o termo '.'(H, T) é uma lista.

O primeiro elemento, chamado cabeça, é H, que é seguida pelo conteúdo do restante da lista, T, também chamado de cauda. A lista [1, 2, 3] seria representada internamente como '.'(1, '.'(2, '.'(3, []))). Um atalho sintático é [H | T], que é mais usado para construir regras. Uma lista pode ser processada como um todo processando o primeiro elemento, e em seguida o restante da lista, de forma recursiva.

4.3.6 Strings

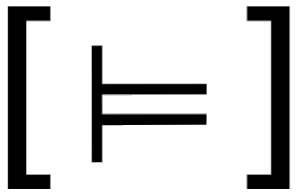
Strings são normalmente escritas como uma seqüência de caracteres entre aspas. É comum serem representadas internamente como listas de códigos de caracteres, em geral utilizando a codificação local ou Unicode, se o sistema dá suporte a Unicode.

5 LINGUAGENS SEMELHANTES

5.1 Logtalk

Logtalk é uma linguagem de programação orientada a objetos baseada em Prolog extendendo e aproveitando a linguagem com encapsulamento de código moderno e mecanismos de reutilização de código sem comprometer os recursos de programação declarativa.

Figura 5 – Logotipo de Logtalk



5.2 Mercury

Mercury é uma linguagem de programação funcional puramente lógica e declarativa desenvolvida inicialmente na Universidade de Melbourne, sua sintaxe é baseada na sintaxe de Prolog e tem grandes influências da linguagem Haskell e também pertence ao paradigma funcional.

Figura 6 – Logotipo de Mercury

5.3 Oz

Oz é uma linguagem de programação multiparadigma que foi influenciada por Prolog, Lisp e Earland criada na universidade de Saarland implementada no *Mozart Programming Sys-*

tem, que possui uma máquina virtual, semelhante ao Java, para que as compilações sejam consistentes e portáveis nas diferentes plataformas.

Figura 7 – Logotipo do Mozart Programming System

6 EXEMPLOS DE PROGRAMAS

Um programa em Prolog é uma base de conhecimento γ. E o programa é invocado através de uma *query* Φ. O valor retornado é o conjunto de ligamentos das variáveis em Φ, se a query sucede, ou falha. O interpretador retorna uma resposta por vez.

Um programa em Prolog pode ser dividido em duas partes, as declarações isto é, a definição da base de conhecimentos γ, e as consultas Φ, que resultam na resposta do programa.

6.1 Hello World

```
1 hello_world :-
2     write('Hello, World!'), nl.
```

6.2 Fatorial

Definimos na primeira linha o **fato** de fatorial, de que o fatorial de 0 é 1. Logo após, criamos as **regras** para o fatorial.

```
1 fatorial(0,1).
2
3 fatorial(N,F) :-
4    N>0,
5    N1 is N-1,
6    fatorial(N1,F1),
7    F is N * F1.
```

Consultas que podemos fazer ao programa:

```
1 ?- fatorial(0,1).
2 Yes
3 ?- fatorial(10,What).
4 What = 3628800
5 Yes
```

Com a primeira linha, perguntamos se o fatorial de 0 é 1. Já na segunda linha queremos perguntar quanto é o fatorial de 10. Com isso obtemos o número em "What".

6.3 Quicksort

Como pode-se ver a declaração do objetivo *pivot* age como a implementação da divisão e conquista do quicksort, separando o arranjo em sub-arranjos de acordo com a necessidade

fazendo a chamada recursiva.

```
1 pivot(_, [], [], []).
2 pivot(Pivot, [Head|Tail], [Head|LessOrEqualThan], GreaterThan):-
3     Pivot >= Head, pivot(Pivot, Tail, LessOrEqualThan, GreaterThan).
4 pivot(Pivot, [Head|Tail], LessOrEqualThan, [Head|GreaterThan]) :-
5     pivot(Pivot, Tail, LessOrEqualThan, GreaterThan).
```

Aqui já vemos o algoritmo, ainda em chamadas recursivas separando os pedaços ordenados e não ordenados, de acordo com o algoritmo do quicksort.

```
1 quicksort([], []).
2 quicksort([Head|Tail], Sorted) :- pivot(Head, Tail, List1, List2),
3     quicksort(List1, SortedList1), quicksort(List2, SortedList2),
4     append(SortedList1, [Head|SortedList2], Sorted).
```

7 CONCLUSÃO

Prolog é uma linguagem de programação com 45 anos de história que até hoje mantém grande relevância nas áreas de interpretação de linguagem natural e principalmente de Inteligência Artificial por ser uma linguagem de sintaxe relativamente simples e pela facilidade de se declarar fatos, regras, relações e objetivos de forma quase natural e sucinta.

Apesar de suas aplicações acontecerem principalmente no ambiente acadêmico continua sendo uma linguagem confiável com grande potencial. Porém não recebe toda a atenção que merece por ter criado as fundações para muitos princípios teóricos em que se baseiam muitas linguagens de programação.

Referências

COLMERAUER, Alain; ROUSSEL, Philippe. The birth of prolog. 1992.

DAVIS, Ernest. **Horn clause logic**. 2003. Disponível em: <https://cs.nyu.edu/courses/spring03/G22.2560-001/horn.html>.

JúNIOR, Ilaim Costa; Sá, Claudio Cesar de. Tutorial de prolog. 2003.

LAGO, SILVIO. Prolog. 1991.

MOURA, Paulo. **LogTalk.org**. 2018. Disponível em: <http://logtalk.org/>.

MOURA, Paulo Jorge Lopes de. **Design of an Object-Oriented Logic Programming Language**. 2003. Tese (Doutorado) — Universidade da Beira Interior.

MOZART.GITHUB.IO. Disponível em: <http://mozart.github.io/>.

SWI-PROLOG.ORG. Disponível em: <http://swi-prolog.org/features.html>.

WIKILIVROS. **Prolog/Noções básicas de Prolog — Wikilivros, Livros abertos por um mundo aberto.** 2013. [Online; accessed 26-setembro-2018]. Disponível em: <https://pt.wikibooks.org/w/index.php?title=Prolog/No%C3%A7%C3%B5es_b%C3%A1sicas_de_Prolog&oldid=250417>.

YESLOGIC; OPTURION; SOMOGYI, Zoltan. **Mercury.org**.