Prolog: Programação em Lógica

Gustavo C. M. Sousa[1], **Braully R. da Silva**[1] **Djean Araújo**[1] **Leonardo Priori**[1]
Ramon V. Silva[1] **Bruno M. de Oliveira**[1]

[1]Instituto de Informática – Universidade Federal de Goiás
74001-970 - Goiânia - GO

{gustavomota, braully, djean, leonardop, ramon, brunomendes}@inf.ufg.br

Abstract. *The choice of a programming language that adequates to the development of a given application is a crucial decision during the process of development. Many languages provide tools that allows an enhancement in the development of some applications. This paper aims at showing the main features of the Prolog programming language, and also discuss about some aspects of programming in logic, and its possible use in the process of development.*

Resumo. *A escolha de uma linguagem de programação adequada para o desenvolvimento de uma determinada aplicação é uma decisão crucial durante o processo de desenvolvimento. Muitas linguagens possuem ferramentas que permitem maior eficiência no desenvolvimento de certas aplicações. Este artigo, visa demonstrar as principais características da linguagem de programação Prolog, além discutir alguns aspectos da programação em lógica, e sua possibilidade de uso no processo de desenvolvimento.*

1. Introdução

Prolog é o mais conhecido representante das linguagens de programação em lógica. Este surgiu como uma linguagem para auxiliar nos projetos de processamento de línguas naturais coordenado por Alain Colmerauer na Faculté des Sciences de Luminy em Marseilles em 1970. Um dos objetivos do projeto era construir um tradutor automático de francês para inglês. Para executar tal tarefa, Alain utilizava uma linguagem de sua criação, chamada de Q-systems.

No entanto essa implementação não era satisfatória e não oferecia recursos necessários para que fosse possível prosseguir sua pesquisa. Também havia um grupo de pesquisadores da área de automação de provas de teoremas que estavam interessados no modelo de inferências adotado por Alain em seu sistema. Então Alain e esse grupo de pesquisadores começaram a trabalhar em uma ferramenta que seria capaz de lidar com essas necessidades. Após dois anos de trabalho, em 1972, esta ferramenta estava pronta, e recebeu o nome de Prolog, como uma abreviação de "*PROgramming in LOGic*". No entanto, esta não era apenas mais uma ferramenta de processamento de línguas naturais ou provas de teoremas, mas sim uma verdadeira linguagem de programação.

Com o tempo a linguagem se espalhou pelas universidades da Europa, e o desenvolvimento de implementações mais eficientes, bem como novas pesquisas no campo, impulsionaram o desenvolvimento da linguagem. No entanto, Prolog ainda não tinha

chegado a atrair muita atenção entre a maioria dos programadores do mundo. Até que em 1981 a decisão do governo japonês em adotar programação lógica em sua Quinta Geração de Computadores, atraiu as atenções para a programação lógica, e principalmenta para a linguagem Prolog. O projeto japônes não foi bem sucedido, mas muitas pesquisas foram realizadas nessa área, e grande parte do desenvolvimento atual da linguagem se deve á esse projeto.

Atualmente Prolog vem sendo utilizado em muitas universidades como uma linguagem de programação relacional, ajudando a incutir nos estudantes uma nova forma de pensar em relação a um problema. Além disso, Prolog também continua sendo utilizado em áreas de inteligência artificial, como processamento de linguagens, sistemas inteligentes, entre outros.

2. Programação Relacional

As linguagems de programação podem ser classificadas de diversas formas. Um método de classificar uma linguagem é de acordo com a visão que ela impõe no usuário. Nesse ponto, as linguagens de programação podem ser classificadas basicamente como procedurais, ou relacionais.

Um usuário de uma linguagem procedural, deve pensar no programa como uma série de instruções que a máquina deve executar, independente da modularização das instruções, ou organização em um nível de abstração maior, em uma linguagem procedural, o programador é o responsável por definir o que o computador irá executar.

Ao contrário das linguagens procedurais, em uma linguagem relacional, a idéia é definir relações existentes entre entidades, ou grupos de entidades. Dessa forma, o programador não precisa se preocupar como o computador irá processar as informações, esta é a tarefa do interpretador/compilador que deve checar as relações existentes e retornar a resposta desejada. Este método permite que o programador tenha uma visão mais afastada em relação á máquina, e mais próxima do pensamento humano.

Prolog é uma linguagem de programação relacional, baseada na lógica de primeira ordem. A lógica de primeira ordem, também conhecida como lógica de predicados, é uma linguagem formal utilizada para representação de conhecimento e solução de problemas. Essa característica permite ao Prolog, a capacidade de representar as mais variadas situações existentes.

3. Características da linguagem Prolog

O objetivo desse artigo não é o ensino da linguagem Prolog em si, e somente dar ao leitor uma visão sobre essa linguagem, e tentar despertar o interesse do mesmo a pesquisar e estudar essa fascinante linguagem de programação. Por esse motivo tentaremos apenas descrever alguns aspectos básicos da linguagem. Para compreensão de alguns conceitos dessa seção se faz necessário um conhecimento mínimo da linguagem da Lógica de Predicados. Ao longo dessa parte demonstraremos alguns exemplos de programas simples em Prolog, e discutiremos seu funcionamento.

A únidade básica de dados em um programa em Prolog é o termo. Existem três diferentes tipos de termos: constantes, variáveis, e compostos. Os termos constantes po-

dem ser átomos, ou números. Átomos são constantes de texto, e são representados por uma seqüência de letras, números e *underscore* que se inicia com uma letra minúscula. Além dos átomos, existe outro tipo constante em Prolog, os números. Termos váriaveis são termos que podem assumir diferentes valores ao longo da execução e são representados por um seqëência de letras, números e *underscore*, que se inicia com uma letra maiúscula. Os valores assumidos por uma variável, só possuem efeito dentro da própria cláusula, pois este é o seu escopo. Por fim, temos os termos compostos, que são termos cujo valor depende do valor de outros termos. Estes termos são representados por um nome e seus parametros.

Um programa em Prolog consiste de uma série de cláusulas conectadas entre si através de relações. Para estabelecer estas relações, são importantes dois conceitos: fatos e regras. Um fato é utilizado para estabelecer uma relação, e é definido pelo seu nome seguido por uma lista de argumentos. Uma regra é por sua vez é um fato especial, cuja validade, depende da validade de outros fatos. Um exemplo de como definir fatos e regras pode ser visto na Figura 1.

```
% Fato definindo João como homem
homem(joao).

% Fato definindo uma relação (amizade) entre Ana e Maria.
amiga(ana,maria).

% João é o progenitor de José
progenitor(joao,jose).

% Regra definindo que para q João seja pai de José
% ele deve ser seu progenitor e ser homem.
pai(joao,jose) :- progenitor(joao, jose), homem(joao).

% Regra mais geral, utilizada para definir todos
% os casos onde X é progenitor de Y e X é homem.
pai(X, Y) :- progenitor(X, Y), homem(X).

% Pergunta para o programa se João é pai de José
? pai(joao,jose).

% Neste caso X assume os valores para os quais João
% é pai de X, ou seja, retorna os filhos de João.
? pai(joao,X).
```

Figura 1: Representações de algumas estruturas em Prolog

No exemplo da Figura 1, foram utilizados os operadores “,” e “:-”, que representam respectivamente os operadores lógicos “e”, e “se”. Além desses existem vários outros símbolos, como o operador “ou”, representado por “;”, o operador de corte “!”, entre outros, que não são usados nestes exemplos.

Outro exemplo, mostrado na Figura 2, é um programa utilizado para calcular o fatorial de um determinado número.

```
fatorial(0,1).

fatorial(N,F) :-
                N > 0,
                N1 is N-1,
                fatorial(N1,F1),
                F is N * F1.

% pergunta por fatorial de 10.
? fatorial(10,F).

% testa se fatorial de 3 é 6
? fatorial(3,6).
```

Figura 2: Programa para calcular o fatorial de um número

O programa para calcular o fatorial, consiste de duas cláusulas, sendo a primeira uma cláusula unitária que indica que o fatorial de 0 é 1, usando o predicado fatorial(0,1). A segunda clausula é uma regra que especifica que o fatorial de N é F se N $>$ 0 e N1 é N-1 e o fatorial de N1 é F1 e F é N*F1. Ou seja, primeiro é testado se N é maior que 0, caso afirmativo, N1 recebe o valor de N-1, e é testado o predicado fatorial(N1, F1). Após este predicado, F1 representa o valor do fatorial de N-1, este é multiplicado por N, e temos o valor do fatorial.

4. Implementações

A primeira implementação do Prolog foi concebida por Colmerauer e Roussel em 1972. Escrito em Fortran, o interpretador rodava de forma sofrível, sendo extremamente lento. Em 1977, Warren e Pereira escreveram um interpretador/compilador para computadores DEC-10, eliminando os maiores problemas em relação a lentidão existente no sistema de Colmerauer. Nesta versão, Warren criou um algoritmo, que mais tarde seria chamado de WAM (Warren Abstract Machine), que tornou-se a base de grande parte das implementações posteriores. Em 1980, McCabe completou sua versão do Prolog para microcomputadores, e após algum tempo Pereira surgiu com uma versão escrita em C para sistemas UNIX.

Logo, várias implementações começaram a surgir, e novas características passaram a ser adicionadas. Tornando as implementações existentes, cada vez mais distantes entre si. Surgiu então a necessidade de estabelecer um padrão entre as diversas implementações existentes de interpretadores/compiladores Prolog.

Em 1984 um grupo britânico começõu a trabalhar no estabelecimento de um padrão para o Prolog. Em 1985 foi formado um grupo pela AFNOR (Association Française de Normalisation) que começou a trabalhar em conjunto com o grupo britânico. Apenas em 1987 foi estabelecido um grupo de trabalho pela ISO/IEC para trabalhar na

padronização. Em 1995 foi estabelecido o padrão para as características básicas das implementações, e apenas em 2000 foi definido o padrão para as partes modulares e características adicionais. Foram estabelecidos padrões para sintaxe e semântica, métodos de I/O e manipulação de exceções.

Segue abaixo uma rápida descrição de algumas implementações existentes para Prolog, e algumas de suas características.

4.1. GNU Prolog

GNU Prolog é uma implementação que suporta compilação nativa de programas escritos em prolog, além de um interpretador interativo. Sua implementação é baseada na WAM, e segue o padrão ISO para Prolog com algumas extensões. GNU Prolog é um software livre e segue a licença GNU, podendo ser livremente distribuído.

Além do suporte a compilação nativa, o GNU Prolog pode ser usado como um interpretador interativo, e compilação para byte-codes.

Características:

- mais de 300 predicados pré-definidos
- extensões para sockets, chamadas de sistema, variáveis globais, entre outros
- interface bi-direcional entre Prolog e C
- grande parte dos predicados pré definidos não usados, não são ligados no processo de linkedição.

Plataformas: GNU/Linux, Win32, SCO, Solaris, FreeBSD, OpenBSD, NetBSD, Darwin (MacOS X), SunOs, Solaris, Irix.

4.2. SWI-Prolog

SWI-Prolog é outro compilador para Prolog, licenciado sob a LGPL, e como o GNU Prolog também é software livre. Este compilador começou a ser desenvolvido em 1987, sendo voltado principalmente para a criação de aplicações reais. Junto com o compilador é possível obter um *toolkit* para programação gráfica.

Características:

- grande conjunto de predicados pré-definidos, incluindo os predicados da primeira parte do padrão ISO, do padrão definido pelo Edinburgh Prolog, e predicados definidos em outras implementações, como Quintus e SICStus Prolog
- interface com C e C++
- *garbage collector* e manipulação de excessões de acordo com o padrão ISO
- bibliotecas para reconhecer analisar arquivos SGML, HTML, XML, RDF, entre outros.

Plataformas: *Unix-like*, Windows (95/98/ME/NT/2000/XP), MacOS X e BeOS.

5. Conclusão

Apesar de ser utilizada principalmente no meio acadêmico, ou em áreas especializadas como Inteligência Artificial, podemos perceber que a linguagem de programação Prolog,

pode ser também utilizada para o desenvolvimento de outros tipos de aplicações, sem perder muitas de suas vantagens.

Durante este artigo foi discutido o surgimento da linguagem Prolog, e seu desenvolvimento nos anos seguintes. Além disso, foi analisado o modelo de programação em lógica, e suas vantagens. Finalmente foi realizada uma análise de algumas implementações existentes, e o processo de padronização da linguagem Prolog.

Referências

Colmerauer, A. and Roussel, P. (1993) “The Birth of Prolog”, The History of Programming Languages Conference, ACM.

Malpas, J. (1986) “Prolog: A Relational Language and its Applications”, Prentice-Hall., p. 1-96.

Casanova, M. (1986) “Programação em Lógica”, Editora Gráfica Formato.

Fisher, J. (1999) “Prolog Tutorial”, http://www.csupomona.edu/~jrfisher/www/prolog_tutorial/contents.html December.

Hodgson, J. (1999) “Prolog: The ISO Standard Documents”, http://pauillac.inria.fr/~deransar/prolog/docs.html, June.

Diaz, D. (2003) “The GNU Prolog web site”, http://gnu-prolog.inria.fr/.