Introdução ao Prolog

Luís Gustavo Werle Tozevich[1], **Jaime Antonio Daniel Filho**[1]

[1]Universidade Federal de Santa Maria (UFSM)

lgtozevich@inf.ufsm.br, jafilho@inf.ufsm.br,

1. Introdução

Prolog é uma linguagem de programação [CLOCKSIN et. al., 2022], amplamente utilizada no campo da inteligência artificial e da linguística computacional. Foi criado na década de 1970 por Alain Colmerauer e Philippe Roussel [NUGUES, 2006]. Prolog é especialmente adequada para problemas que envolvem bancos de dados relacionais, lógica matemática, entendimento de linguagem natural, entre outros, como exemplificado por CLOCKSIN et. al. (2022). Ao contrário das linguagens de programação imperativas tradicionais, como C ou Java, segundo NUGUES (2006) Prolog é baseada em lógica de predicados, permitindo a definição de fatos e de regras que o sistema pode usar para inferir novas informações e resolver problemas complexos.

A principal vantagem do Prolog reside em sua capacidade de manipular símbolos e de executar buscas lógicas de maneira eficiente. Tal característica confere-lhe uma vantagem significativa na resolução de problemas complexos de forma automática e eficaz. Com uma abordagem declarativa, é possível concentrar-se na formulação do problema, delegando ao motor de inferência de Prolog a tarefa de encontrar as soluções apropriadas.

Neste estudo, discorrer-se-á acerca dos fundamentos da linguagem Prolog, suas principais características e as áreas de aplicação nas quais ela manifesta maior eficácia. Ademais, analisar-se-ão exemplos práticos e a estrutura básica de um programa em Prolog, proporcionando uma visão abrangente para aqueles que buscam compreender e utilizar esta linguagem de programação.

2. A linguagem

Prolog distingue-se das linguagens de programação convencionais, tanto na sua estrutura quanto na sua filosofia de resolução de problemas. Enquanto as linguagens tradicionais, como C ou Java, adotam uma abordagem imperativa onde o programador especifica passo a passo as instruções que o computador deve executar, Prolog segue um paradigma declarativo. Nesta abordagem, o programa é descrito em fatos e em relações entre esses fatos [BRATKO, 2001], o que diferencia-se dos paradigmas procedurais em que o foco está nos passos para resolver um problema [BRATKO, 2001]. Nesta linguagem, pergunta-se quais relações são verdadeiras e quais objetos ocorrem no problema. Por isso, o Prolog é visto como uma linguagem descritiva e prescritiva [CLOCKSIN et. al., 2022].

Escrever um programa em Prolog envolve a especificação de fatos, que são afirmações básicas sobre o domínio do problema [NUGUES, 2006], e regras, que são declarações condicionais que permitem a derivação de novos fatos a partir dos já conhecidos, ou seja, é dizer que um fato depende de uma série de outros fatos [CLOCKSIN et. al., 2022]. Ademais, realiza-se consultas ao sistema em busca de respostas baseadas nos fatos e nas regras definidas [STERLING; SHAPIRO, 1994].

Por conseguinte, ao adotar uma abordagem declarativa, Prolog proporciona uma maneira intuitiva de modelar problemas em termos de suas propriedades e de suas relações fundamentais. Essa característica faz de Prolog útil em áreas como processamento de linguagem natural, resolução de problemas abstratos, raciocínio automático e diversas áreas de inteligência artificial, onde a capacidade de raciocinar automaticamente sobre o conhecimento é crucial.

3. Distinção entre dados

Em Prolog, não existem tipos de dados, como nas linguagens de programação tradicionais. Todos os dados são de um único tipo, o termo, porém diferenciam-se em outros elementos dependendo

da forma em que são declarados.

- **Variáveis** são sempre iniciadas com uma letra maíuscula ou um underscore e funcionam como uma espécie de incógnita matemática, cujo valor é inicialmente desconhecido e, uma vez descoberto, não poderá ser alterado.
- **Átomos** são símbolos únicos que iniciam com uma letra minúscula ou estão envolvidos por aspas.
- **Números** são representados por uma sequência de dígitos os quais podem ser acompanhados por '.' para número reais, '-' para números negativos e 'e' para notações científicas.
- **Termos compostos** são formados por um átomo, chamado de 'functor', seguido por uma lista contida entre parênteses de termos separados por vírgula, os parâmetros. O número de parâmetros é chamado de aridade.

4. Fatos e regras

Um programa em Prolog consiste em uma série de cláusulas que afirmam se algo é verdadeiro. Uma cláusula é descrita na forma

```
Cabeçalho :- Corpo.
```

e é lida como "se Corpo é verdadeiro, então Cabeçalho é verdadeiro".

Chamamos de fatos as cláusulas com corpos vazios. Por exemplo,

```
humano(piveta).
```

E chamamos de regras as cláusulas com corpo definido, como

```
humano(piveta) :- true.
```

Os dois exemplos acima são equivalentes, ou seja, afirmam que piveta é humano.

5. Regras complexas

As regras precisam garantir, frequentemente, que vários requisitos sejam verdadeiros ou que somente um dos vários requisitos seja verdadeiro. Para isso, é possível fazer o uso de predicados, que são estruturas que encapsulam um determinado comportamento e permitem que esse comportamento seja aplicado a parâmetros específicos. O Prolog fornece alguns predicados embutidos, como ',/2', ';/2', './2' e '\+/1'.

A vírgula (,) é utilizada para a conjunção lógica, permitindo a especificação de múltiplas condições que devem ser verdadeiras para que a regra como um todo seja considerada verdadeira.

O ponto e vírgula (;) é usado para expressar a disjunção lógica, permitindo a alternância entre múltiplas opções. Ele indica que uma condição ou outra deve ser verdadeira para que a regra como um todo seja considerada verdadeira.

O ponto (.) é empregado para criar listas em Prolog, atuando como um construtor de lista incremental. Por exemplo, a lista [1, 2, 3] pode ser representada como .(1, .(2, .(3, []))), onde [] denota a lista vazia e cada . adiciona um elemento à lista.

Os predicados ',/2', ';/2' e './2' possuem dois parâmetros. O predicado '\+/1', por sua vez, possui somente um único parâmetro e fornece a negação como falha, ou seja, a condição será verdadeira se nenhuma prova da regra original for encontrada.

Um exemplo de uma regra complexa em Prolog é a declaração humano(X) :- pensa(X), existe(X). Neste exemplo, humano/1 é o predicado, onde /1 indica que ele possui um parâmetro. Essa declaração representa a relação de 'ser humano', onde um indivíduo X é considerado humano se ele pensar (pensa(X)) e existir (existe(X)).

```
pensa(piveta).
existe(piveta).

humano(X) :- pensa(X), existe(X).

?- humano(piveta).
```

6. Execução de um programa

A execução de um programa em Prolog inicia-se com a formulação de uma consulta pelo usuário, representando a pergunta ou objetivo a ser alcançado. Em seguida, o mecanismo Prolog emprega sua lógica de inferência para encontrar uma solução que satisfaça a consulta. Essa metodologia se destaca pela sua característica de busca de refutação, na qual o sistema busca mostrar a impossibilidade da negação da consulta, confirmando, portanto, sua validade. Quando uma solução é encontrada, as atribuições de variáveis resultantes são então relatadas ao usuário. A estratégia de execução do Prolog, conhecida como retrocesso cronológico, permite a exploração de múltiplas alternativas para alcançar a solução desejada. Em caso de falha em uma alternativa, o sistema retrocede para tentar outras possibilidades até encontrar uma solução válida. Esta abordagem reflexiva e iterativa é fundamental para o funcionamento eficiente dos programas em Prolog.

Por exemplo, considere o seguinte programa em Prolog que define algumas relações pessoais:

```
filho(pedro, joao).
filho(luis, joao).
filho(marcos, pedro).
filho(henrique, marcos).
filho(jaime, luis).
filho(guilherme, luis).

descendente(X, Y) :- filho(X, Y).
descendente(X, Y) :- filho(X, Z), descendente(Z, Y).
```

No programa acima, os fatos 'filho(X, Y)' estabelecem relações de filiação, indicando que X é filho de Y. A regra 'descendente(X, Y)' é definida de forma recursiva: um indivíduo X é descendente de outro Y se X for filho direto de Y ou se houver uma cadeia de descendência que conecte X a Y através de outros indivíduos. Um exemplo de consulta seria solicitar 'descendente(X, pedro)', em que o Prolog irá buscar todas as pessoas que são descendentes de pedro. Essa consulta retornará X = marcos; X = henrique.

7. Conclusão

Este estudo proporcionou uma compreensão abrangente da linguagem de programação Prolog, enfatizando suas características distintivas e sua abordagem baseada em lógica de predicados para resolver problemas. Em contraste com abordagens imperativas tradicionais, delegando ao mecanismo de inferência a responsabilidade pela busca de soluções. Desse modo, a capacidade de Prolog em realizar inferências lógicas e em manipular dados consolida sua importância como uma ferramenta valiosa em diversas áreas acadêmicas e profissionais.

Referências

BRATKO, Ivan. Prolog Programming for Artificial Intelligence. 3. ed. Harlow: Addison-Wesley, 2001.

CLOCKSIN, William F.; MELLISH, Christopher S. Programming in PROLOG. Springer Science & Business Media, 2003.

NUGUES, Pierre M. An introduction to prolog. Springer Berlin Heidelberg, 2006.

STERLING, Leon; SHAPIRO, Ehud. The Art of Prolog: Advanced Programming Techniques. 2. ed. Cambridge: MIT Press, 1994.