Universidade de Itaúna

Isis Estevan Mendonça

Pedro Campos

LINGUAGENS DE PROGRAMAÇÃO LÓGICA

Itaúna

2022

INTRODUÇÃO:

Antes de explicar Programação Lógica vamos entender sua base

O sentido da programação lógica é trazer o estilo da lógica matemática à programação de computadores. Matemáticos e filósofos encontram na lógica uma ferramenta eficaz para desenvolvimento de teorias.A lógica proporciona uma maneira de demonstrar se uma questão é verdadeira ou falsa.
Então a forma particular de lógica simbólica usada para programação lógica é chamada
de cálculo de predicados de primeira ordem ou cálculo de predicados

Temos as proposições que são representadas por termos simples, constantes ou variáveis.

Proposições podem ser definidas de dois modos: um no qual a proposição é definida
como verdadeira e outro no qual a verdade da proposição é algo que deve ser determinado. Em outras palavras, as proposições podem ser fatos ou consultas.Proposições compostas têm duas ou mais proposições atômicas ou simples, conectadas por conectores lógicos, ou operadores.

As lógicas são melhores em sua forma
simples; isso significa que a redundância deve ser minimizada.
Um problema do cálculo de predicados, como descrevemos até agora, é que existem
muitas maneiras de definir proposições com o mesmo significado; ou seja, existe uma
grande quantidade de redundância. Esse não é um problema para especialistas em lógica
matemática, mas se o cálculo de predicados será usado em um sistema automatizado
(computadorizado), é um problema sério. Para simplificar as coisas, é desejável uma
forma padrão para proposições. A forma clausal, uma forma relativamente simples de
proposições, é uma delas. Todas as proposições podem ser expressas em forma clausal.
Uma proposição em forma clausal tem a seguinte sintaxe geral:

$$B_1 \cup B_2 \cup \ldots \cup B_n \subset A_1 \cap A_2 \cap \ldots \cap A_m$$

Linguagens de programação lógica:

Linguagens usadas para programação lógica são chamadas de linguagens declarativas, porque os programas escritos nelas consistem em declarações, em vez de atribuições e sentenças de fluxo de controle. Essas declarações são na verdade sentenças, ou proposições, em lógica simbólica.

Uma das características essenciais das linguagens de programação lógica é sua semântica, chamada de semântica declarativa. O conceito básico dessa semântica é que
existe uma maneira simples de determinar o significado de cada sentença, e ele não depende de como ela poderia ser usada para resolver um problema.Por
exemplo, o significado de uma proposição em uma linguagem de programação lógica
pode ser concisamente determinado a partir da própria sentença

A programação em uma linguagem de programação lógica é não procedural. Os programas em tais linguagens não descrevem exatamente como um resultado será computado, mas a forma do resultado. A diferença é o fato de assumirmos que o sistema
de computação pode, de alguma forma, determinar como o resultado será computado.

PROLOG

Prolog (Programação Lógica) é uma linguagem de programação que se enquadra no paradigma de Programação em Lógica Matemática.
Foi criada em meados de 1972 por Alain Colmerauer e Philippe Roussel, baseados no conceito de Robert Kowalski da interpretação procedimental das cláusulas de Horn(Uma cláusula de Horn é uma disjunção lógica dos literais, onde no máximo um dos literais é positivo, e todos os outros são negativos.). A motivação para isso veio em parte da vontade de reconciliar o uso da lógica como uma linguagem declarativa de representação do conhecimento com a representação procedimental do conhecimento, que era popular na América do Norte no final da década de 1960 para início de 1970.
Muito do desenvolvimento moderno do Prolog veio dos projetos de computadores da quinta geração (FGCS), que desenvolveu uma variante do Prolog chamada Kernel Language para seu primeiro sistema operacional.

Apesar do longo tempo de desenvolvimento, Prolog ainda não é uma linguagem portável, já que cada implementação usa rotinas completamente diferentes e incompatíveis entre si. Por exemplo, um programa trivial que faz um loop de ler uma linha da console e escreve a

mesma linha, terminando quando for entrada uma linha vazia, é impossível de ser escrito de forma que qualquer interpretador consiga rodar.

História:

A linguagem de programação Prolog nasceu de um projeto que não tinha por foco a implementação de uma linguagem de programação, mas o processamento de linguagens naturais. Na Universidade de Marselha, Alain Colmerauer e Robert Pasero trabalhavam na parte de linguagem natural e Jean Trudel e Philippe Roussel trabalhavam na parte de dedução do projeto. Interessado pelo método de resolução SL, Trudel persuadiu um dos seus inventores, Robert Kowalski, que se uniu ao projeto. O projeto resultou em uma versão preliminar da linguagem Prolog em fins de 1971 sendo que a versão definitiva apareceu em fins de 1972.Desde então tem sido utilizada para aplicações de computação simbólica, como banco de dados relacionais, compreensão de linguagens naturais (português, inglês, etc.), automação de projetos, análise de estruturas bioquímicas e sistemas especialistas. Como podemos ver, o Prolog se tornou uma referência quando se trata de linguagem de programação voltada para inteligência artificial e lingüística computacional.

Características:

O Prolog é uma linguagem declarativa, ou seja, ao invés de o programa estipular a maneira de chegar à solução passo-a-passo, como acontece nas linguagens procedimentais ou orientadas a objeto, ele fornece uma descrição do problema que se pretende computar utilizando uma coleção de fatos e regras (lógica) que indicam como deve ser resolvido o problema proposto. Como podemos ver, o Prolog é mais direcionado ao conhecimento do que aos próprios algoritmos.

Além de ser uma linguagem declarativa, outro fato que o difere das outras linguagens é a questão de não possuir estruturas de controle (if-else, do-while, for, switch) presentes na maioria das linguagens de programação. Para isso utilizamos métodos lógicos para declarar como o programa deverá atingir o seu objetivo.

Um programa em Prolog pode rodar em um modo interativo, o usuário poderá formular queries utilizando os fatos e as regras para produzir a solução através do mecanismo de unificação.

Conceitos Básicos:

Os tipos de dados comumente existentes em outras linguagens, não são em empregados ao Prolog. Todos os dados são tratados como sendo de um único tipo, conhecido como termo, que pode ser uma constante, uma variável ou um termo composto

Fatos:

Em Prolog são fornecidos os fatos e as regras para uma base de dados, que posteriormente serão executadas consultas (queries) em cima da base de dados.

A estrutura de um fato é formada por um predicado, seus argumentos (objetos) e finalizamos a instrução com um ponto(.) equivalente ao ponto-vírgula das linguagens comuns de programação. Veja a seguir:
predicado(argumento1,argumento2...).
O predicado é a relação sobre os quais os objetos irão interagir.

Exemplos:
amiga(joana, maria).
Definimos uma relação de amizade entre dois objetos, joana e maria.

homem(jose).
Note que quando usamos apenas um objeto, o predicado passa a ser uma característica do próprio objeto.

Então apenas para concluir, é importante ressaltar que os nomes dos predicados e dos objetos devem sempre começar por letra minúscula como devem ter notado nos exemplos anteriores. Os predicados são escritos primeiro, seguido pelos objetos que são separados por vírgula. Também é importante lembrar de que a ordem dos objetos poderá interferir no resultado de uma aplicação.

Questões:

A sintaxe das questões varia de acordo com os compiladores existentes, mas basicamente é um fato antecedido de um ponto de interrogação ou o comando apropriado para o tipo de compilador. Por exemplo:
?- amiga(joana,maria).

Quando uma questão é feita, o Prolog realiza uma busca na sua base de conhecimento, procurando um fato que seja igual ao da questão. Se o fato for encontrado é retornado "YES", caso contrário o programa retornará a saída "NO".

Variáveis:

No Prolog uma variável não é um contêiner cujo valor pode ser atribuído como ocorre nas outras linguagens. Inicialmente uma variável é uma incógnita, cujo valor é desconhecido, mas quando instanciamos um objeto a ela, a mesma não poderá ser mais modificada. Vejamos um exemplo.

Primeiramente forneceremos uma seqüência de fatos para a base de conhecimento:

gosta(joão,flores).
gosta(joão,maria).
gosta(paulo,maria).

Em seguida realizaremos uma questão:
?- gosta(joão,X).

O que acontece quando compilarmos essa instrução?

Inicialmente a variável X encontra-se com um valor desconhecido (não-instanciado), o Prolog procura então por um fato na base de conhecimentos que se iguale a questão, ou seja, ele irá procurar algum fato que tenha o mesmo predicado, o mesmo número de argumentos e que o primeiro argumento seja "joão". É importante saber que a busca é realizada na ordem em que os fatos foram passados. Então no nosso exemplo, o objeto flores será atribuída à variável X.

Para concluir, devo lembrar que as variáveis devem começar sempre com letra maiúscula ou o caractere underscore. É importante lembrar que uma vez que uma variável é instanciada, ela não poderá mudar o objeto.

Conjunção e Disjunção:

Para montarmos uma lógica mais específica nas nossas questões, podemos utilizar os conceitos de conjunção e disjunção. Equivalente ao "AND" e o "OR" de outras linguagens, podemos realizar uma consulta mais avançada separando os fatos por vírgulas no caso de conjunção (AND), ou ponto e vírgula no caso de disjunção (OR).

Dados os seguintes fatos:
amiga(joana,maria).
amiga(clara,maria).

Podemos realizar a seguinte questão:
?- amiga(joana,X),amiga(clara,X).

No exemplo anterior fornecemos dois fatos, e realizamos uma questão cruzando-os. Como maria é amiga de joana e carla, a variável X receberá o objeto Maria, e a questão retornará "YES".

Agora vamos ver um outro exemplo aproveitando os mesmo fatos:
?- amiga(joana,X);amiga(roberta,X).

Nesse exemplo utilizamos o operador de disjunção (OR), ou seja, se uma das questões coincidirem com os fatos, será retornada "YES". No nosso exemplo, será retornado "YES" e o X será atribuído ao objeto "maria", pois apresentam o mesmo predicado, o mesmo número de objetos, e o primeiro objeto, é o mesmo do existente na base de dados.

A conjunção (e) é escrita como ",", enquanto a disjunção (ou) é escrita como ";"

Regras:

Utilizamos as regras para fazermos a construção de questões complexas. Especificamos situações que podem ser verdadeiras se algumas condições forem satisfeitas.

Para utilizarmos uma regra, usamos o símbolo ":-" que indica uma condição (se). Vamos exemplificar o uso da regra com um exemplo simples.

Dados os fatos:
pai(arthur,silvio).
pai(arthur,carlos).
pai(carlos,xico).

pai(silvio,ricardo).

Utilizaremos a seguinte regra:
avo(X,Z) :- pai(X,Y), pai(Y,Z).

Isso significa que se alguém é pai de uma pessoa, que por sua vez é pai de outra pessoa, então ele é avô. Vamos realizar uma querie para conferir a regra:

?- avo(arthur,xico),avo(arthur,ricardo).

Retornará a saída: "YES"

Ou seja, "arthur" é avô de "xico" e "ricardo", pois "arthur" é pai de "silvio" e "carlos", que por sua vez são pais de "xico" e "ricardo".

Operadores:

Podemos utilizar operadores para construirmos regras ainda mais específicas, no Prolog existem tanto os operadores relacionais quanto os aritméticos.

Entre os operadores relacionais temos:
Igualdade: =
Diferença: \= (em alguns compiladores o operador de diferença é "<>")
Menor que: <
Maior que: >
Menor ou igual: =< (alguns compiladores seguem a versão ">=")
Maior ou igual: >=

Vamos construir um pequeno exemplo com operadores relacionais para verificar se o número passado é positivo ou negativo.

Para isso, construiremos a seguinte regra:
positivo(numero) :- numero > 0.

Em seguida realizaremos uma consulta:
?- positivo(2).
Que retornará "Yes".

Agora vamos aos operadores ariméticos. São eles:

+, -, *, /, mod, is.

O operador é utilizado quando quisermos obter o módulo, e o operador is é utilizado para obtermos um resultado.

Verifique a seguinte regra:
o_dobro(Numero,Result) :- Result is Numero * 2.

Realizaremos a questão:
?- o_dobro(5,X).

Obteremos o resultado:
X = 10

É importante lembrar que uma variável terá que receber o valor da operação aritmética através do operador "is"

Entrada e Saída:

Já vimos como atribuir os fatos, como construir regras e fazer questões sobre elas, mas tudo que recebemos como retorno, foi um "Yes" ou "No". Assim como nas outras linguagens, o Prolog também possui propriedades de entrada e saída de dados, são eles read() e write().

Vamos ver um pequeno exemplo:
ola :- read(X), write("Olá "), write(X).

Faremos a chamada:
?- ola. "Luciano".

Note que a regra não possuiu nenhum parâmetro, portanto colocamos apenas o nome da regra sem atributos, e em seguida o dado a ser lido pelo comando read(X), lembrando que cada instrução sempre é finalizada com um ponto(.).

É importante ressaltar que dados equivalentes ao tipo "String" de outras linguagens, devem sempre ser utilizados entre apóstrofos, caso contrário, quando digitássemos o nome "Luciano", o compilador iria aceitar a entrada como uma variável não alocada, e imprimiria um valor estranho na saída.

Negação:

Tipicamente, uma consulta é avaliada como falsa no caso de não estar presente nenhuma regra positiva ou fato que dê suporte ao termo proposto. Isso é chamado hipótese do mundo fechado; assume-se que tudo o que é importante saber está na base de dados, de modo que não existe um mundo exterior que pode possuir evidências desconhecidas. Em outras palavras, se um fato não é conhecido ser verdadeiro (ou falso), assume-se que ele é falso.

Operadores de Controle:

Backtracking:

Backtracking é um procedimento dentro da linguagem Prolog. Uma busca inicial em um programa nesta linguagem segue o padrão Busca em profundidade (depth-first search), ou seja, a árvore é percorrida sistematicamente de cima para baixo e da esquerda para direita. Quando essa pesquisa falha, ou é encontrado um nó terminal da árvore, entra em funcionamento o mecanismo de backtracking. Esse procedimento faz com que o sistema retorne pelo mesmo caminho percorrido com a finalidade de encontrar soluções alternativas.

Exemplo:

Considerando uma base de dados família, fazemos a seguinte consulta:

?- pai(roberto,X), mae(vera,X)

O compilador tenta satisfazer o primeiro objetivo. Quando conseguir, tenta satisfazer o segundo. Caso não consiga, ele retorna ao ponto onde encontrou a solução para o primeiro objetivo (backtracking).

Comando Cut:

O comando cut permite indicar ao Prolog quais sub-objetivos já satisfeitos não necessitam ser reconsiderados ao se realizar um backtracking. Isto é, ele aborta o processo de backtracking. O uso do comando cut é importante porque permite que o programa rode mais rápido, sem perder tempo com sub-objetivos que não contribuem para determinar a resposta do objetivo principal. Além disso, o programa ocupará menos memória, pois não será necessário armazenar todos os sub-objetivos considerados (pontos do backtracking). Em alguns casos, o cut evita que o programa entre em laço infinito.

Comando Fail:

Inversamente ao comando cut, o predicado pré-definido fail sempre falha. O operador de corte pode ser combinado com o predicado fail para produzir uma falha forçada. Uma conjunção de objetivos da forma

cabeça :- objetivo₁, ..., objetivo_n, !, fail.

é usada para informar ao PROLOG: se a execução chegou até esse ponto, então pode abandonar a tentativa de satisfazer a regra. A conjunção falha devido ao fail, e o objetivo-pai falha devido ao corte.

Conclusões:

Como podemos ver, o Prolog é uma linguagem muito poderosa, principalmente na área de Inteligência Artificial onde é líder absoluta. Mostramos os principais comandos da linguagem, mas vale lembrar que vimos apenas o básico do Prolog, existem muitas outras propriedades não abordadas no artigo, e muitas implementações diferentes da linguagem, cada um com o seu próprio padrão. Entre as implementações do Prolog, podemos citar o Visual Prolog (Turbo Prolog), o SWI Prolog, GNU Prolog, Amzi! Prolog, entre muitas outras já existentes.

Bibliografia:

https://pt.wikipedia.org/wiki/Prolog

http://www.linhadecodigo.com.br/artigo/1697/descobrindo-o-prolog.aspx

SEBESTA, Robert W. Conceitos de Linguagem de Programaçã. 11ª edição . Bookman;27 abril 2018.