Paradigmas de Programação

Paradigma Lógico com Prolog

Introdução

Este material contém notas de aula sobre o *Paradigma Lógico*. O objetivo é apresentar os conceitos deste paradigma utilizando a linguagem de programação Prolog.

Paradigma Lógico

As linguagens do paradigma lógico também fazem parte do paradigma declarativo. Relembrando, o paradigma declarativo:

- É oposto ao paradigma imperativo;
- Descreve/declara o que um programa faz, e não o seu fluxo de controle/como ele funciona.
- Descreve seus resultados desejados sem listar os passos ou a lista de comandos necessários para isso.
- Concentra-se em detalhes do problema e abstrai os detalhes de implementação.
- Dentro deste paradigma estão o paradigma lógico e o paradigma funcional.

O paradigma lógico é fundamentado na lógica formal, especialmente na utilização de **fatos**, **regras** e **inferências** que envolvem o domínio em questão.

- **Fato**: sentença aceita como verdadeira que estabelece um relacionamento entre objetos. Exemplos:
    - Todo ser vivo é mortal.
    - Brasil é um país.

    - Blumenau é uma cidade.
    - `<tr>` é uma tag HTML.
    - A guerra dos cem anos durou mais de cem anos.
- **Regra**: declaração que estabelece relações entre predicados. As regras definem como novas informações podem ser inferidas a partir dos fatos existentes. Exemplos:
    - João é pai de Otávio.
    - Otávio é pai de Pedro.
    - ∴ João é avô de Pedro.
    - ∴ Pedro é neto de João.
    - Fulano governa a Esbórnia.
    - Esbórnia é um país.
    - ∴ Fulano governa um país.
- **Inferência** (ou **consulta**): pergunta feita ao sistema lógico para deduzir ou inferir informações com base nos fatos e regras definidos. Em linguagens de programação lógica, uma consulta geralmente é uma expressão que solicita ao sistema para encontrar soluções para determinadas relações. Exemplos:
    - Brasil é um país?
    - Capão do Leão é uma cidade?
    - Existe um presidente chamado Fulano?
    - Quem é o presidente da Esbórnia?
    - Existe número maior que 11?

> Cálculo de predicados é a notação usada nas linguagens de programação lógica atuais. A programação em linguagens de programação lógica não é processual. Os programas nessas linguagens não indicam exatamente como um resultado deve ser calculado, mas descrevem a forma e/ou características necessárias do resultado. O que é necessário para fornecer esta capacidade em linguagens de programação lógica é um meio conciso de fornecer ao computador as informações relevantes e um processo de inferência para calcular os resultados desejados. O cálculo de predicados fornece a forma básica de comunicação ao computador, e o método de prova, denominado resolução, desenvolvido inicialmente por Robinson (1965), fornece a técnica de inferência (SEBESTA, 2012, tradução nossa).

Linguagens de Programação Lógica

A primeira linguagem de programação lógica que se tem conhecimento é a linguagem Absys (*Aberdeen System*) (ELCOCK, 1990). Hoje, Prolog é a linguagem lógica mais conhecida e utilizada, servindo também como base para outras linguagens como AnsProlog, Datalog e Mercury. Outra linguagem lógica digna de registro é a CLIPS, voltada para a implementação de sistemas especialistas (WATKIN; ADAM; PERUGINI, 2019).

Aplicações

O paradigma lógico é visto em aplicações de

- Inteligência Artificial (ex: robótica)
- SGBDs (Sistemas de Gerenciamento de Bancos de Dados)
- Sistemas Especialistas (WATKIN; ADAM; PERUGINI, 2019)
- Processadores de Linguagem Natural
- Demonstração de Teoremas
- Construção de Compiladores.

A página do SWI-Prolog traz exemplos reais de aplicações desenvolvidas com Prolog.

A Linguagem Prolog

Prolog significa *programming in logic* (programação em lógica). Esta linguagem surgiu na década de 70 e, até os dias de hoje, segundo Sebesta (2018), é a única linguagem lógica amplamente utilizada.

Um programa em Prolog é formado por um conjunto de declarações chamadas de **cláusulas**[1] (fatos e regras). O conjunto de cláusulas é chamado de **base de conhecimento**. Com uma base de conhecimento criada, é possível realizar consultas por meio de uma interface.

[1] Por vezes, cláusulas são chamadas também de *relações*.

A base de conhecimento é acessada por meio do **motor de inferências**. Este motor é um componente que aplica as regras aos fatos, parando a execução quando encontra a solução – que pode estar implícita – ou quando não há regra a ser aplicada.

Para maior precisão nos resultados obtidos, a base de conhecimento precisa ser o mais completa possível, pois em Prolog adota-se a **hipótese de mundo fechado**. Isso significa que o que não é explicitamente verdadeiro é assumido como falso, ou, em outras palavras, falso é aquilo que não é comprovadamente verdadeiro. Assim, a base de conhecimento representa o universo completo de informações disponíveis, e qualquer coisa fora desse universo é considerada falsa.

Prolog possui implementações disponíveis para praticamente todas as plataformas de computador populares (SEBESTA, 2012). Existem implementações compiladas e interpretadas desde a década de 70.

Ferramentas

Em sistemas *Unix-like*, a ferramenta utilizada é o `swipl` (SWI-Prolog), também executada pela palavra reservada `prolog`.

O ambiente `SWISH` (*SWI-Prolog for SHaring*) é uma alternativa *online* para programação com Prolog, e pode ser acessado através do endereço https://swish.swi-prolog.org/.

- Clicando em *Program* é exibida uma tela dividida em três partes: base de conhecimento, consultas e resultados.
- A base de conhecimento conterá o conjunto de fatos e regras. Por padrão, fica na área à esquerda.
- A área de consultas é realizada para se fazer perguntas à base de conhecimento, verificando se ela pode provar que a consulta é verdadeira. Por padrão, fica na área à direita, na parte inferior.
- A área de resultados exibe os resultados das consultas. Por padrão, fica na área à direita, na parte superior.

Funcionamento

Prolog funciona da seguinte maneira:

1. O usuário interage com uma interface realizando uma consulta
2. A interface passa a consulta ao motor de inferência
3. O motor de inferência consulta a base de conhecimento
4. Após a consulta, o motor possui uma resposta e a repassa à interface
5. A interface exibe a resposta para o usuário

A Figura 1 exibe um esquema gráfico resumindo o funcionamento da linguagem, descrito acima.

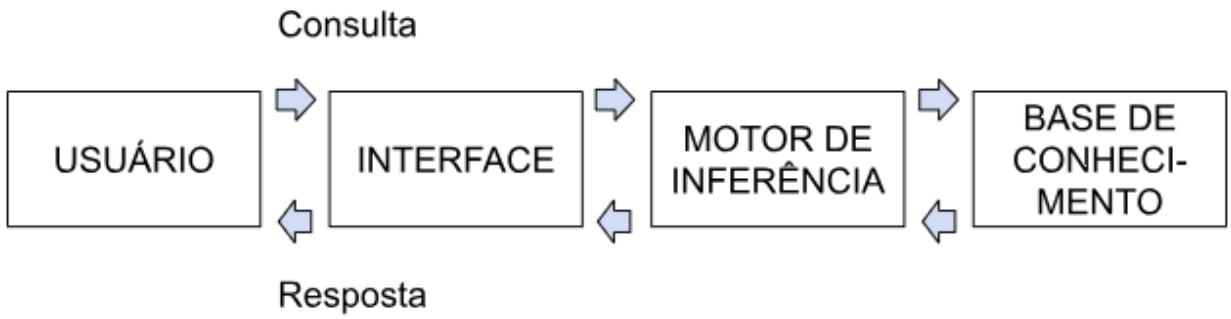

Figura 1. Esquema de consulta e resposta em um sistema Prolog.

Sintaxe

A linguagem Prolog possui uma sintaxe rigorosa.

- Fatos e regras são escritos iniciando com letras minúsculas.
- Variáveis são escritas com letra maiúscula. Costuma-se utilizar `X`, `Y`, `Z` etc.
- O caractere ponto final (`.`) é o último caractere da declaração, tendo a função de terminador.
- Qualquer expressão que apresentar espaço em branco ou caractere(s) especial(is) deve estar entre aspas simples.
- Aspas duplas são permitidas para delimitar *strings*.

CUIDADO AO COPIAR OS EXEMPLOS

É comum (e válido!) copiar os exemplos do material para reproduzi-los. No entanto, é necessário cuidar a cópia dos caracteres de aspas, pois eles podem causar erros de sintaxe.

Os caracteres ', ‘ e ’ são todos caracteres de aspas simples, mas são interpretados de maneira diferente pelo interpretador Prolog. O primeiro caractere de aspa simples (') é o único reconhecido para delimitar as expressões.

Em relação às aspas duplas, apenas o caractere " é usado para delimitar *strings*. Se um caractere diferente do padrão ASCII para as aspas duplas for usado, como “ (aspa dupla curvada para dentro) ou ” (aspa dupla curvada para fora), o interpretador Prolog não reconhecerá corretamente o delimitador da *string* e resultará em um erro de sintaxe.

Portanto, caso ocorra erro de sintaxe gerado pelo uso das aspas simples ou duplas, sugere-se a digitação do caractere correto manualmente.

Além disso, comentários em Prolog são precedidos pelo símbolo “`%`”. Não há suporte nativo para comentários de múltiplas linhas.

Elementos da Linguagem

Os exemplos a seguir trazem fatos em Prolog. Cada fato está acompanhado de um comentário que explica o seu significado em linguagem natural.

```
lp(prolog). % Prolog é uma linguagem de programação.
taghtml(tr). % tr é uma tag HTML.
cidade(blumenau). % Blumenau é uma cidade.
cidade(gaspar). % Gaspar é uma cidade.
estado('santa catarina'). % Santa Catarina é um estado.
sigla(sc, 'santa catarina'). % SC é uma sigla de Santa Catarina.
país(brasil). % Brasil é um país.
país(esbórnia). % Esbórnia é um país.
presidente(lula, brasil).
presidente(fulano, esbórnia). % Fulano é presidente da Esbórnia.
```

Todos esses fatos podem ser inseridos em um arquivo de texto com extensão `.pl` para serem testados no `swipl` ou acrescentados na base de conhecimento da ferramenta *online* `SWISH`.

Nos exemplos acima, `lp`, `taghtml`, `cidade`, `estado`, `sigla`, `país` e `presidente` são os **fatos** ou **predicados**.

Da lógica matemática advém o conceito de **átomo**, o qual também é adotado pela linguagem Prolog. Segundo Meidanis (2017),

> "Um átomo indica um objeto ou uma relação. Nomes de objetos como `maria`, `livro`, etc. são átomos. Nomes de átomos sempre começam com letra minúscula. Nomes de predicados são sempre atômicos também. Os grupos de caracteres `?-` (usado em perguntas) e `:-` (usado em regras) são também átomos".

No âmbito da matemática, um predicado é uma função cujo contradomínio é um conjunto formado por dois elementos: verdadeiro e falso. Assim, quando uma consulta for feita utilizando um predicado, o resultado será verdadeiro ou falso.

O exemplo a seguir traz uma regra em Prolog:

```
neto(X,Y) :- pai(Y,Z), pai(Z,X). % Define a relação "neto" em termos da relação "pai".
```

A definição de uma regra segue sempre desta forma. A regra é criada à esquerda, seguida de :-, indicando a relação lógica que será estabelecida. Em seguida, são especificadas as condições que devem ser atendidas para que a conclusão da regra seja válida. Em outras palavras, "uma regra possui cabeça e corpo, no formato cabeça :- corpo" (GOMES E SOUZA, 2024). Ainda de acordo com Gomes e Souza (2024), interpreta-se a regra da seguinte forma: *"se o corpo é verdadeiro, então a cabeça é verdadeira*".

No exemplo acima, a regra `neto(X, Y)` é definida indicando o que é necessário para que `X` seja neto de `Y`. Para que essa relação seja verdadeira, duas condições devem ser atendidas: `Y` deve ser pai de `Z` e `Z` deve ser pai de `X`. A vírgula utilizada para separar as cláusulas serve como operador de conjunção lógica (E lógico). Já o ponto e vírgula tem significado de operador de disjunção lógica (OU lógico).

A quantidade de parâmetros que compõem uma cláusula é chamada de **aridade**. Assim, as relações `neto` e `pai` têm aridade 2.

Com a declaração da regra, é possível identificar em uma base de conhecimento, através de uma **consulta**, se `X` é neto de `Y`. A consulta (ou pergunta) é feita utilizando os caracteres ?- sucedidos pela cláusulas que se deseja verificar. Por exemplo, para verificar se João é neto de Pedro, a consulta seria:

```
?- neto(joão, pedro).
```

Tanto o `SWISH` quanto o `swipl` já inserem automaticamente os caracteres `?-` em seus *prompts*, sendo necessário apenas digitar o restante.

É importante observar que a regra `neto` repete as variáveis `X`, `Y` e `Z` ao longo da cláusula. De acordo com Gomes e Souza (2024), uma variável tem validade apenas dentro da cláusula onde se encontra. Assim, duas ocorrências de `X` (e qualquer outra variável) na mesma cláusula correspondem à mesma variável.

Operadores

Os principais operadores em Prolog são:

- Aritméticos
    - Adição (`+`)
    - Subtração (`-`)
    - Multiplicação (`*`)
    - Divisão (`/`)
- Relacionais
    - Igual a (`==`)
    - Diferente de (`\==`)
    - Maior que (`>`)
    - Menor que (`<`)
    - Maior ou igual a (`>=`)
    - Menor ou igual a (`=<`)
- Lógicos

- E lógico (`,`)
- OU lógico (`;`)
- Negação de prova (`\+` ou `not`)

Exemplos

Esta seção mostra exemplos para reprodução nas ferramentas `SWISH` e `swipl`.

1. **Criação da base de conhecimento**. O conteúdo do quadro abaixo será a base de conhecimento:

```
cidade(blumenau).
cidade(pelotas).
cidade(riogrande).
cidade(brusque).
cidade(florianopolis).
cidade(criciuma).
cidade(joinville).
cidade(indaial).
cidade(timbo).
cidade(taio).
cidade(fraiburgo).
cidade(itajai).
cidade(gaspar).
cidade(maua).
cidade(pomerode).
cidade(biguacu).
cidade(bombinhas).
cidade(camboriu).
cidade(itapema).
```

   1.1. A base deve ser inserida no quadro à esquerda do `SWISH` ou em um editor de texto simples. Ao optar pelo editor de texto, o arquivo deve ser salvo com a extensão `.pl` (exemplo: `cidades.pl`).

2. **Carregamento da base**. Este passo não é necessário no SWISH. Para carregar a base de conhecimento no `swipl`, é necessário executar a ferramenta a partir do diretório em que o arquivo `.pl` está salvo e, do ambiente `swipl`, carregar o arquivo digitando seu nome entre colchetes. Caso o nome do arquivo seja `cidades.pl`, ele será carregado digitando: `[cidades].`
3. **Realização de consultas**. No SWISH, basta digitar no campo destinado às consultas (quadro no canto inferior direito), e no ambiente swipl, basta digitar no prompt. Nas duas ferramentas os caracteres `?-` já vêm inseridos, bastando digitar a(s) cláusula(s) a ser(em) consultada(s). Exemplos:
   3.1. `?- cidade(capão do leão).`
   3.2. `?- cidade(joinville).`
   3.3. `?- cidade(X).`

Qual é o problema da consulta 3.1?
Como se interpreta a consulta 3.2?
Qual foi o resultado da consulta 3.3?

Uma consulta deve retornar um valor lógico (`true` ou `false`) ou uma instância que a satisfaça, caso uma ou mais variáveis seja usada na consulta. No exemplo 3.3, a consulta solicita algum valor para o qual `X` satisfaça a relação `cidade`. Como há muitos valores possíveis, o `SWISH` disponibiliza, entre outros, um botão "`Next`", podendo exibir mais valores que satisfaçam a relação. No `swipl` isto é possível digitando "`;`"

O `swipl` é encerrado teclando "`halt.`".

O arquivo `times.pl` contém uma base de conhecimento. Analisando esta base, como realizar consultas que equivalham às perguntas abaixo?

1. Que time tem verde na bandeira?
2. O Grêmio tem vermelho na bandeira?

As respostas seriam:

1. `?- bandeira(X, verde).`
2. `?- bandeira(gremio, vermelho).`

Na primeira resposta, um time que possui verde na bandeira será exibido. Caso existam mais times, é possível solicitar que o próximo time seja informado, até que não existam mais valores que possam ser vinculados à variável. Neste caso, utiliza-se a variável `X` onde seria escrito um time, assim o motor de inferência procura na base de conhecimento um átomo válido para substituir aquela variável.
Na segunda consulta, basta utilizar os átomos `gremio` e `vermelho`, pois deseja-se saber se a bandeira do Grêmio tem vermelho.

Alguns exemplos de uso de operadores (podem ser testados no `SWISH` ou no `swipl`):

```
?- not(abacaxi==banana).
?- not(abacaxi\==banana).
?- \+(abacaxi==banana).
?- -3 > 2.
?- 4+4^2 =< 24.99.
```

Os arquivos `pai.pl` e `politico.pl` contêm mais exemplos contendo fatos, regras e operadores já explicados neste material.

Prolog permite o uso de um "curinga" no texto. Ele é representado pelo underline (_). Neste caso, qualquer coisa digitada no lugar será válida. Por exemplo, tendo o fato abaixo na base de conhecimento:

```
bairro(_).
```

Ao realizar consultas inserindo qualquer valor na cláusula `bairro`, o retorno sempre será `true`. Exemplos:

```
?- bairro(araraquara).
?- bairro(ricardo).
?- bairro(abcdefgh).
```

Observação: não é necessário inserir os caracteres `?-`, eles são inseridos automaticamente pelas ferramentas.

Tipos

A linguagem Prolog possui tipagem dinâmica e fraca. Tipagem dinâmica significa que os tipos das variáveis são verificados em tempo de execução, não em tempo de compilação. Isso permite uma maior flexibilidade no desenvolvimento, pois as verificações de tipo são realizadas durante a execução do programa. Exemplo:

```
?- X = 5.
?- X = 5.0.
?- X = "Hello, World!".
?- X = [figueirense, chapecoense, joinville].
```

Ao digitar as quatro consultas acima, a variável `X` é instanciada com diferentes tipos de dados em cada consulta: primeiro com um número inteiro, depois com um número real, depois com uma *string* e, por fim, com uma lista.

A tipagem fraca pode ser exemplificada pela cláusula `atom_concat`, que concatena dois átomos formando um terceiro. Exemplo:

```
?- atom_concat('Número: ', 42, Resultado).
```

O resultado exibido será `Resultado = 'Número: 42'`, ou seja, Prolog permitiu a concatenação entre os dois átomos, mesmo que eles sejam, respectivamente, uma expressão textual (delimitada por aspas simples) e um valor numérico.

Mais informações sobre tipos de dados em Prolog podem ser consultados na página sobre tipos de dados, do SWI-Prolog.

Desvantagens

Entre as desvantagens da linguagem Prolog, podem ser citadas:

- A hipótese de mundo fechado, pois Prolog consegue provar o que é verdade, mas não prova o que é falso.

- Limitações intrínsecas do modelo declarativo: por se preocupar em declarar o que o programa deve fazer, e não como fazer. Por exemplo: como tornar eficiente um processo de ordenação?
- Problemas com o operador de negação (`not`), que não é equivalente à negação lógica.

Para mais detalhes sobre as desvantagens, sugere-se a leitura da Seção 16.7 de Sebesta (2012).

Listas + Recursão + Funtor + Unificação + Backtrack + E/S

Os tópicos e conceitos em vermelho estarão presentes em versões futuras do material.

Conclusão

Este material abordou alguns dos conceitos fundamentais do Paradigma Lógico na linguagem de programação Prolog. O Paradigma Lógico, inserido no contexto do paradigma declarativo, enfatiza a descrição do que um programa faz, em vez de abordar como ele funciona. Ele é baseado na lógica formal e abrange predicados, regras e inferências, que constituem a base do domínio em questão.

Prolog tem sido a ferramenta mais popular desse paradigma. Sua estrutura fundamental consiste em um conjunto de cláusulas, compreendendo fatos e regras, que juntos formam a base de conhecimento do sistema. Essa base de conhecimento é acessada e consultada por meio de um motor de inferências, que utiliza a técnica de resolução para deduzir informações a partir dos fatos e regras definidos.

Além disso, o paradigma lógico e a linguagem Prolog têm aplicações variadas, desde inteligência artificial e sistemas especialistas até processadores de linguagem natural e demonstração de teoremas. A sintaxe rigorosa e a estrutura das cláusulas em Prolog facilitam a expressão de problemas em termos lógicos, permitindo uma abordagem mais próxima do domínio do problema. No entanto, é importante estar ciente das limitações da linguagem, como a hipótese de mundo fechado, que pode levar a resultados inesperados em certas situações.

Exercícios

1) Qual é a aridade de `atom_concat`?

2) Pesquise e formule uma resposta que explique corretamente por que a consulta abaixo, na coluna da esquerda, resultou na saída indicada na coluna da direita.

Entrada	**Saída**
`?- functor(cidade(N, E), X, Y).`	`X = cidade,` `Y = 2`

3) Resolva os exercícios da Lista #4.

Referências

ELCOCK, Edward W. Absys: the first logic programming language — A retrospective and a commentary. The Journal of Logic Programming, v. 9, n. 1, p. 1-17, 1990.

GOMES E SOUZA, Rodrigo Rocha. Prolog: conceitos e exercícios básicos. Disponível em: https://rodrigorgs.github.io/aulas/mata56/aula02-prolog. Acesso em: 8 mar. 2024.

MEIDANIS, João. MC346 - Paradigmas de programação: Prolog. Disponível em: https://www.ic.unicamp.br/~meidanis/courses/mc346/2017s2/prolog/apostila-prolog.pdf. Acesso em: 7 mar. 2024.

SEBESTA, Robert W. Concepts of Programming Languages. Pearson. 10th ed. 2012.

SEBESTA, Robert W. Concepts of Programming Languages. Pearson. 12th ed. 2018.

SWI-Prolog. Datatypes. Disponível em: https://www.swi-prolog.org/datatypes.html. Acesso em: 8 mar. 2024.

WATKIN, Jack L., VOLK, Adam C., PERUGINI, Saverio. An introduction to declarative programming in clips and prolog. In: World Congress in Computer Science, Computer Engineering, and Applied Computing, 2019. Disponível em: https://ecommons.udayton.edu/cgi/viewcontent.cgi?article=1178&context=cps_fac_pub. Acesso em: 16 ago. 2024.