Disciplina de Linguagens de Programação
2ª Lista de Exercícios
Curso de Engenharia de Computação
UEMG Ituiutaba
https://github.com/mauro-hemerly/UEMG-2019-1

Assunto: Predicados, Fatos, Regras, Cláusulas, operadores aritméticos e de comparação

Uma **base de conhecimento** Prolog é formada por **cláusulas** finalizadas por ponto (.).

Uma cláusula pode ser um **fato** ou uma **regra**. Uma regra possui **cabeça** e **corpo**, no formato **cabeça :- corpo**.

Identificadores começados por letras minúsculas são átomos. Identificadores começados por letras maiúsculas são variáveis.

Uma variável tem validade apenas dentro da cláusula onde se encontra. Assim, duas ocorrências de X na mesma cláusula correspondem à mesma variável:

```
% Se X for joao e pensa(joao) é verdadeiro,
% concluímos que existe(joao).
existe(X) :- pensa(X).
```

Por outro lado, é possível usar o mesmo nome em cláusulas diferentes para se referir a coisas diferentes:

```
% Podemos ter X = joao na primeira cláusula
% e X = maria na segunda cláusula.
humano(X) :- homem(X).
humano(X) :- mulher(X).
```

Note que duas variáveis diferentes na mesma cláusula podem ter o mesmo valor:

```
amigo(joao, maria).
amigo(joao, jose).
amigo(joao, joao).

% Qual o resultado das consultas a seguir?
% amigo(X, X).
% amigo(X, Y).
```

Variáveis começadas por _ não são exibidas no resultado da consulta. Exemplos: $\_X$, $\_Y$. Usamos nomes começados por _ para variáveis cujo valor não estamos interessados.

Exemplo de código:

```
turma(mata56, turma1, rodrigo).
turma(mata56, turma2, rodrigo).
turma(mata62, turma3, ivan).
turma(mata62, turma4, rodrigo).

%% Consulta: Quais sao as disciplinas de Rodrigo (não exibe as turmas)?
% turma(D, _T, rodrigo).
```

Além disso, a variável anônima, _ (somente _), tem a seguinte peculiaridade: cada ocorrência dela representa uma variável distinta.

Exemplo de consulta:

```
%% Quais são os professores (não importa a disciplina e a turma)?
% turma(_, _, P).
%% Note que o primeiro e o segundo argumentos de turma
%% são variáveis diferentes, que podem assumir valores
%% diferentes. A consulta é equivalente a
% turma(_X, _Y, P).
```

Ao escrever termos, **,** representa e (conjunção lógica) e **;** representa ou (disjunção lógica). O predicado **not/1** representa negação; ou seja, **not(X)** é verdadeiro se e somente se X for falso.

É possível comparar dois termos com os predicados binários == (igual) e \ == (diferente). Você pode escrever no formato de predicado comum, ==(X, Y) ou na notação infixa: X == Y. Exemplos de consultas (execute para ver o resultado):

```
X == X.
X == Y.
amigo(X, X), X == joao. % considere a base de amigos
amigo(X, X), X == maria. % considere a base de amigos
```

Exercícios

1. Quantos **fatos**, **regras**, **cláusulas**, e **predicados** podemos encontrar na seguinte base de conhecimento?

```
mulher(vincent).
mulher(mia).
homem(jules).

pessoa(X) :- homem(X); mulher(X).
ama(X,Y) :- pai(X,Y).
pai(Y,Z) :- homem(Y), filho(Z,Y).
pai(Y,Z) :- homem(Y), filha(Z,Y).
```

Aritmética em Prolog

O que acontece ao executar a seguinte consulta em Prolog?

```
?- X = 1 + 2.
```

Era de se esperar que X fosse instanciado com o valor 3, mas não é isso o que acontece. Em vez disso, X é instanciado com o termo complexo **1+2** ou, equivalentemente, **+(1, 2)**.

```
% As duas formas são idênticas:
1 + 2 == +(1, 2).
```

Operador is

O operador binário **is** é similar ao **operador de unificação**, =, com uma diferença: o lado direito (e apenas o lado direito) é interpretado como uma expressão aritmética e seu resultado é calculado antes da unificação.

Exemplos:

```
?- X is 1 + 2.
?- 3 is 1 + 2.
```

A segunda consulta retorna true porque o lado direito do operador is é avaliado como expressão aritmética, resultando no valor 3, e 3 (lado esquerdo) pode ser unificado com 3 (lado direito).

Contra-exemplo:

```
?- 1 + 2 is 3.
```

O último exemplo não funciona porque o lado esquerdo do is não é avaliado como expressão aritmética. Ao tentar unificar +(1, 2) com 3, a unificação falha.

Operadores aritméticos

O Prolog possui os operadores mais comuns, como +, **-**, ***, /, **abs**, **sin**, **cos**, **tan**, **exp**, **ln**, **log**, **sqrt**, dentre outros. A precedência entre operadores é respeitada.
Exemplo:

```
?- X is 1 + 2 * 3.
?- X is sqrt(9).
```

Não confunda operadores com predicados! Operadores aritméticos são funções que retornam um número.

Operadores de comparação

Prolog admite operadores de comparação, como > e <. Ao usar um desses operadores, ambos os lados são avaliados como expressão aritmética.
Exemplos:

```
?- 4 > 3.   /* true */
?- 8 + 1 < 5 + 5.   /* true */
?- X is +(8, 1), X < X + 1.   /* true */
```

Operadores de comparação:

```
<: menor que
>: maior que
>=: maior ou igual a
=<: menor ou igual a (a maioria das outras linguagens usa <=)
=:=: igual a
=\=: diferente de
```

Note que esses operadores **não realizam unificação**! Por isso, as variáveis devem estar instanciadas antes de se realizar a comparação.
Exemplo:

```
?- X =:= 5.
/* [Error] =:=/2: Arguments are not sufficiently instantiated */
```

Qual o resultado das seguintes consultas?

```
?- 1 + 2 == +(1, 2).
?- 1 + 2 == 2 + 1.
?- 1 + 2 == 1 + X.
?- 1 + 2 = 1 + X.
?- 1 + 2 = X.
?- 1 + 2 = 2 + 1.
?- 1 + 2 = 2 + X.
?- 1 + 2 =:= 2 + 1.
?- 1 + 2 =:= 2 + X.
?- X is 1 + 2.
?- 1 + 2 is X.
```

Exercícios

2. Escreva uma regra para o predicado **entre(X, A, B)**, que indica que o número X está entre os números A e B, inclusive (isto é, deve retornar verdadeiro também se X = A ou X = B). Assuma que A é menor ou igual a B.

3. Reescreva a regra da questão anterior de forma que funcione para quaisquer dois valores inteiros de A e B, seja A maior ou menor do que B.

4. Escreva uma regra para o predicado **triplo(X, Y)**, que indica que X é o triplo do número Y. Teste com a seguinte consulta: **triplo(X, 4)**. O que acontece se você consultar triplo(12, Y), para tentar obter um terço de 12?

5. Escreva uma regra para o predicado **par(X)**, que indica se o número inteiro não-negativo X é par. Lembre-se de que 0 é par. Use o operador **mod(X, Y)**, que retorna o resto da divisão de X por Y.

6. Reescreva a regra da questão anterior sem usar o operador **mod**.

7. Escreva uma regra para o predicado **fatorial(X, Y)**, que indica que o fatorial de X é Y. Dica: considere que o fatorial de 0 é 1 e que o fatorial de X é igual a X multiplicado pelo fatorial de X - 1.

8. Escreva uma regra para o predicado **sucessor(X, Y)**, que indica que o número X é sucessor do número Y.

Exercícios

Considere a seguinte base de conhecimento composta de predicados **progenitor/2**:

```
progenitor(maria, jose).
progenitor(joao, jose).
progenitor(joao, ana).
progenitor(jose, julia).
progenitor(jose, iris).
progenitor(iris, jorge).
```

Considere que o predicado **progenitor(A, B)** significa que A é progenitor (i.e., pai ou mãe) de B.

9. Desenhe a árvore genealógica representada pela base de conhecimento.

10. Escreva uma consulta para responder à seguinte pergunta: "Ana é progenitora de Jorge?"

11. Escreva uma consulta para retornar os progenitores de Íris.

12. Escreva uma consulta para retornar os progenitores de José.

13. Escreva uma consulta para retornar todos os pares **progenitor/filho** da base de conhecimento.

14. Escreva uma consulta para retornar todos os avós de Jorge. Dica: sua consulta será formada por dois termos separados por vírgula.

15. Escreva uma consulta para retornar todos os netos de João.

16. Escreva uma consulta para retornar todos os progenitores comuns de José e Ana.

    Considere a seguinte base de conhecimento composta de predicados **progenitor/2**, **masculino/1** e **feminino/1**:

```
progenitor(maria, jose).
progenitor(joao, jose).
progenitor(joao, ana).
progenitor(jose, julia).
progenitor(jose, iris).
progenitor(iris, jorge).

masculino(joao).
masculino(jose).
masculino(jorge).
feminino(maria).
feminino(julia).
feminino(ana).
feminino(iris).
```

17. Pode-se definir o predicado **filho/2** como sendo o inverso de **progenitor/2**: se X é progenitor de Y, então Y é filho de X. Escreva uma regra para computar o predicado **filho/2** e teste com algumas consultas.

18. Escreva regras para os predicados **mãe/2** e **pai/2**. Teste sua regra.

19. Escreva regras para os predicados **avô/2** e **avó/2**. Teste sua regra.

20. Escreva uma regra para o predicado **irmã/2**. Teste sua regra. Em particular, teste com a consulta **irmã(X, iris)**.

    Considere uma mesa com a seguinte configuração de pessoas:

```
joao      maria      jose      julia      jorge      ana      iris
```

    Isto é, João está imediatamente à esquerda de maria, que está imediatamente à esquerda de José, e assim por diante.

    Nos exercícios a seguir, sempre crie consultas para testar a base de conhecimento.

21. Considere o **predicado a_direita_de(X, Y)**, indicando que X se senta imediatamente à direita de Y. Escreva uma base de conhecimento com esse predicado para representar a configuração de pessoas da mesa.

22. Escreva uma regra para o predicado **a_esquerda_de/2**, que é o inverso de **a_direita_de/2**.

23. Escreva uma regra para o predicado **sao_vizinhos_de(Esq, Dir, Meio)**, que indica que Esq e Dir são os vizinhos à esquerda e à direita de Meio, respectivamente.

24. Escreva uma regra para o predicado **adjacente(X, Y)**, que indica se X e Y estão sentados um ao lado do outro.

25. Escreva uma regra para o predicado **esta_na_ponta(X)**, que indica que X está em uma das cabeceiras da mesa. Dica: mesmo quem está na cabeceira é vizinho de alguém.