# Lista 11

## Prática de laboratório - Fatos e regras

Ferramenta online para realizar consultas: `swish.swi-prolog.org/`

1. Crie uma base de dados em Prolog referente à cadeia alimentar vista em sala de aula usando os predicados `come(X,Y)`, que significa que `X` come `Y`, `animal(X)`, que significa que `X` é um animal, e `planta(X)`, que significa que `X` é uma planta.

```
% urso, peixe, raposa, veado e grama são constantes

%come(X,Y): X come Y
come(urso,peixe)
come(urso,raposa)
come(veado,grama)

%animal(X): X é animal
animal(urso)
animal(peixe)
animal(raposa)
animal(veado)

%planta(X): X é planta
planta(grama)
```

   a) Usando a base da cadeia alimentar criada, faça uma consulta para verificar se um determinado elemento é animal.
   Exemplo de consulta

   > Consulta: `?animal(urso).`
   > Esta consulta retorna se o fato é verdade ou não.

   b) Faça uma consulta para verificar quais elementos o `urso` come.
   Exemplo de consulta

   > Consulta: `?come(urso,X).`
   > Esta consulta retorna quais elementos casam com `X` para tornar o predicado verdadeiro.

   c) Faça uma consulta para encontrar o animal que come planta.

   d) Faça uma consulta para encontrar o animal que come outro animal.

   e) Uma *presa* pode ser definido como um animal que é comido por outro animal na natureza. Crie a regra `presa(X)` na base usando os predicados já existentes. Faça consultas para testar a sua regra.

f) Um *predador* pode ser definido como um animal que come outro animal na natureza. Crie a regra `predador(X)` na base usando os predicados já existentes. Faça consultas para testar a sua regra.

RECORRÊNCIA

As regras em Prolog são condicionais. Os antecedentes (corpo da regra) podem depender de fatos (como as regras feitas até agora), mas também podem depender da própria regra, de modo que a regra pode ser definida em termos de si mesmo. Uma definição na qual o item sendo definido é, ele próprio, parte da definição é chamada uma **definição recorrente**.

Incremente a sua base como mais dados como os mostrados a seguir:

```
%come(X,Y): X come Y
come(urso,peixe)
come(peixe,peixinho)
come(peixinho,alga)
come(guaxinim,peixe)
come(urso,guaxinim)
come(urso,raposa)
come(raposa,coelho)
come(coelho,grama)
come(urso,veado)
come(veado,grama)
come(lince,veado)

%animal(X): X é animal
animal(urso)
animal(peixe)
animal(peixinho)
animal(guaxinim)
animal(coelho)
animal(raposa)
animal(veado)
animal(lince)

%planta(X): X é planta
planta(grama)
planta(alga)
```

Vamos definir uma relação `na-cadeia(X.Y)` que significa *"Y está na cadeia alimentar de X"*. Isso significa uma entre duas coisas:

a) x come y diretamente, ou

b) x come alguma coisa que come alguma coisa ... que come y.

O caso b) pode ser reformulado da seguinte maneira:
b)' x come z e y pertence à cadeia alimentar de z.

A regra pra a definição do item a) é simples de fazer em função do predicado `come(X,Y)`. Por outro lado, a definição do item b) somente funciona com o uso conjunto de definição feita em a), caso contrário, nos remete a um caminho de comprimento infinito de alguma coisa comendo alguma coisa comendo alguma coisa e assim por diante. Definições recorrentes precisam de informações específicas de quando parar.

2. Baseado na ideia de definição recorrente, crie a regra de Prolog para `na-cadeia(X,Y)` usando os predicados existentes na base. (resolvido)

```
na-cadeia(X,Y) :- come(X,Y).
na-cadeia(X,Y) :- come(X,Z), na-cadeia(Z,Y).
```

   Faça consultas na base para testar a regra.

3. Usando o predicado `progenitor(X,Y)`, que significa que `X` é progenitor de `Y`, represente em Prolog os progenitores da sua família (pais, avós), esta será a sua base de dados.
   a) Faça consultas que recupere quem são os filhos de uma determinada pessoa.
   b) Como consultar quem é o pai do pai de uma pessoa? (resolvido)

      Consulta: `?progenitor(Y,fulano),progenitor(Avo,Y).`
      Esta consulta faz uso de uma variável compartilhada, onde o sistema primeiro resolve `progenitor(Y,fulano)`, obtendo um valor para `Y`, em seguida, resolve `progenitor(Avo,Y)` substituindo `Y` obtido na resolução anterior. A resposta buscada será o valor de `Avo`.

   c) Outro tipo de variável importante é a variável *anônima*. Ela é usada quando seu valor específico for irrelevante numa determinada consulta ou definição de regra. Faça uma consulta para saber quem tem filhos. (resolvido)

      Consulta: `?progenitor(X,_).`
      Como neste caso, o nome dos filhos é uma informação irrelevante, a segunda variável pode ficar anônima.

4. Usando o predicado `mulher(X)`, que significa que `X` é mulher, represente as pessoas que são mulheres da sua base. Da mesma forma, com o predicado `homem(X)` represente as pessoas que são homens da sua base.

5. Usando o predicado `casal(X,Y)`, que significa que `X` é casado ou casada com `Y`, represente as pessoas que formam um casal na sua base.

6. Crie uma regra que represente que uma pessoa é ascendente de outra, ou seja, que uma pessoa é o ancestral de outra. Este regra usa a definição de recorrência, onde um ascendente pode ser um progenitor direto X de outra pessoa Y ou X é o progenitor de alguém que é ascendente de Y. Faça consultas para testar esta regra.

7. Crie as regras `irmao(X,Y)` e `irma(X,Y)`, onde `X` é irmão/irmã de `Y` respectivamente.

8. Crie as regras `cunhado(X,Y)` e `cunhada(X,Y)`, onde `X` é cunhado/cunhada de `Y` respectivamente.

9. As cláusulas de Horn são no formato $h \leftarrow p_1, p_2$ onde a vírgula representa o conectivo $\wedge$. No entanto, em Prolog é possível representar o corpo da cláusula com o conectivo $\vee$ usando (;). Para mostrar esta possibilidade, crie a regra `tiotia(X,Y)`, onde `X` é tio ou tia de `Y`.

10. Codifique as regras equivalentes às seguintes sentenças:
    a) Todo mundo que tem filhos é feliz.
    b) Um casal com filhos é formado por duas pessoas que tem filhos em comum.