# Resolução de Problemas em Prolog

[1] Marcelo Ruan Moura Araújo e [2] André Ribeiro Brito
[1 2] Universidade Federal de São Carlos - UFSCar
[1 2] Departamento de Computação
[1] Biomedical Image Processing Group - BIP Group
marcelo.araujo@ufscar.br e andrerbrito@ufscar.br

## 1. Problemas

1. Dada uma lista *Lin* que contém elementos de qualquer tipo, possivelmente com repetições,construir outra lista Lout que mostre quantas vezes cada elemento atômico (átomo, número, lista vazia) aparece na lista dada, inclusive nas sublistas. A lista *Lout* deve conter pares de elementos sendo o primeiro elemento do par um elemento atômico que aparece na lista dada e o segundo elemento do par, o número de vezes que esse elemento aparece na lista. Variáveis e estruturas devem ser descartadas e não serão usadas na contagem.

Por exemplo, dada a lista:

```
Lin = [a,b,Z,x,4.6,[a, x],[],[5,z,x],[]]
```

Deve ser construída a lista

```
Lout=[[a,2],[b,  1],[4.6,1],[x,   3],[[],2],[5,1],[z,1]]
```

2. Dada uma lista *Lin* com elementos de qualquer tipo, construir a codificação Lout dessa lista em que repetições consecutivas de elementos devem ser substituídas por pares da forma[N,E], onde N é o número de repetições consecutivas do elemento E.

```
Lin = [a,a,a,a,b,c,c,a,a,d,e,e,e,e]
```

Deve ser construída a lista

```
Lout = [[4,a],[1,b],[2,c],[2,a],[1,d][4,e]]
```

3. Dada uma lista Lin que seja a codificação de outra lista no formato do exercício anterior, construir a decodificação dessa lista codificada.
Por exemplo, dada a lista:

```
Lin = [[4,a],[1,b],[2,c],[2,a],[1,d][4,e]]
```

Deve ser construída a lista:

```
Lout = [a,a,a,a,b,c,c,a,a,d,e,e,e,e]
```

## 2. Soluções:

Para a execução do exercício proposto foi criado um predicado sem argumentos *cria_lista* para inicializar o procedimento *cria_lista*. Esse predicado chama o

predicado *read* , que lê a lista que o usuário digita e a atribui à variável Lista . O predicado *cria_lista* , que possui 2 argumentos é então chamado, sendo que no primeiro argumento é passada a lista que se deseja processar (variável Lista , que armazena a lista lida), e uma variável Resultado para armazenar o resultado do processamento. Após isso, a lista Resultado é impressa ao usuário.

Para executar a operação de criar uma lista com o formato proposto, foram necessários outros dois predicados auxiliares: *conta_ocorr* e *retira_elem*.

A seguir, o código do *cria_lista* e dos predicados auxiliares:

```
conta_ocorr(E,[],0):!.
conta_ocorr(E,[E],1):!.
conta_ocorr(E,[E|Y],S): conta_ocorr(E,Y,S1), S is S1 + 1, !.
conta_ocorr(E, [X|Y], S):E\==X,conta_ocorr(E,Y,S).

retira_elem(E, [], []):!.
retira_elem(E,[E|Y],L):retira_elem(E,Y,L),!.
retira_elem(E,[X|Y],[X|L]):E\==X,retira_elem(E,Y,L).

cria_lista:write('Digite a lista'),
            read(Lista),
            cria_lista(Lista, Resultado),
            write('Lista resultante:'),
            write(Resultado),
            nl.
cria_lista([],[]).
cria_lista([X|Y],[[X,N]|L]):conta_ocorr(X,Y,N1), N is N1 + 1, retira_elem(X,Y,Z),
cria_lista(Z,L)
```

A lógica utilizada foi a seguinte:

```
cria_lista([],[]).
```

Uma lista vazia resulta em uma lista vazia.

```
cria_lista([X|Y],[[X,N]|L]):conta_ocorr(X,Y,N1),    N    is    N1    +    1,
retira_elem(X,Y,Z),cria_lista(Z,L).
```

Se a lista não for vazia, ela tem um padrão [X|Y], ou seja, ela tem pelo menos um elemento e uma cauda (que pode ser vazia). O elemento é representado pela variável X. Uma lista com esse padrão resultará em uma lista com o padrão [[X,N]|L], onde a variável X representa o elemento atual e N representa a quantidade de vezes que esse elemento foi encontrado na lista lida. L representa a cauda da lista.

Isso será verdade se for verdade que a contagem de ocorrência do elemento X na cauda Y da lista resultar em N1. Como a contagem do elemento foi feita na cauda da lista é necessário somar 1, pois não se considera o primeiro elemento na contagem da cauda. Portanto N is N1 +1. Para garantir que o elemento não seja contado novamente, após ter sido feita a contagem desse elemento ele é retirado da cauda, e uma nova lista é gerada, a lista Z. Então, por fim, a lista Z é passada pro

predicado cria_lista para que seu primeiro elemento seja colocado na lista resultante (que é a cauda da lista resultante anterior) e sejam colocados todos.

O predicado auxiliar conta_ocorr é o responsável por contar o número de vezes que um dado elemento aparece em uma lista. No programa principal, em que ele é usado como auxiliar, o elemento passado é o primeiro elemento da lista e a lista onde deve contar a ocorrência é a sua cauda.

```
conta_ocorr(E,[ ],0):!.
```

O número de vezes que um elemento aparece em uma lista vazia é 0.

```
conta_ocorr(E,[E],1):!.
```

O número de vezes que um elemento E aparece em uma lista com o padrão [E] é 1.

```
conta_ocorr(E,[E|Y],S): conta_ocorr(E,Y,S1), S is S1 + 1, !
```

O número de vezes que um elemento E aparece em uma lista com o padrão [E|Y], ou seja, o primeiro elemento da lista unifica com o elemento E, é S; se for verdade que o número de vezes que o elemento E aparece na cauda da lista é S1 e S is S1 + 1.

```
conta_ocorr(E, [X|Y], S):E\==X,conta_ocorr(E,Y,S)
```

Caso o primeiro elemento da lista não unifique com o elemento E, seja diferente desse elemento, é necessário contar a ocorrência do elemento E na cauda da lista. Essa contagem resultará em S, que será o valor S da contagem na lista principal.

O predicado auxiliar `retira_elem` é o responsável por retirar um elemento de uma lista. Isso é feito para garantir que no ato de se criar a lista não ocorra repetições de elementos na lista resultante.

```
retira_elem(E, [], []):!.
```

Retirar um elemento de uma lista vazia resulta em uma lista vazia.

```
retira_elem(E,[E|Y],L):retira_elem(E,Y,L),!.
```

Retirar um elemento E de uma lista com o padrão [E|Y], onde o primeiro elemento da lista unifica com o elemento E, resultará em uma lista L se for verdade que retirar o elemento E da cauda da lista resultar em L.

```
retira_elem(E,[X|Y],[X|L]):E\==X,retira_elem(E,Y,L).
```

Retirar um elemento E de uma lista onde o primeiro elemento é diferente de E resultará em uma lista com o padrão [X|L], onde X é o elemento diferente de E (ele não deve ser retirado da lista) e uma cauda L, se for verdade que retirar o elemento E da cauda Y da lista original resultar em L.

Resultados de Execução:

```
? cria_lista([[1],1,2,3,1,4,2,4,[1,[2]],[1],[1,[2]],[1],[4]], R).
R = [[[1], 3], [1, 2], [2, 2], [3, 1], [4, 2], [[1, [...]], 2], [[4], 1]] .

? cria_lista([1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1], R).
R = [[1, 20]]
```