Trabalho 03

Prof. Chauã Queirolo

Exercício 1

Construir um predicado em Prolog, denominado "`elem_repetidos`", o qual relaciona uma lista de itens com uma sub-lista (possivelmente vazia) de todos os itens da lista original que aparecem repetidos em qualquer numero de vezes. Seu comportamento é o expresso abaixo:

elem_repetidos

```
?- elem_repetidos([a,b], Z).
Z = [] ?
yes

?- elem_repetidos([a,a,b], Z).
Z = [a] ?
yes

?- elem_repetidos([a, b, c, c, d, e, f, f, g, h, c, c, c, e,
   e, a], Z).
Z = [a, c, e, f] ?
yes
```

Exercício 2

Construir um predicado em Prolog, denominado "`intercalada`", o qual relaciona duas listas de itens (`L1` e `L2` – possivelmente vazias) com uma terceira lista (`L3`), onde esta última contém a intercalação dos elementos das duas outras listas (`L1` e `L2`). Caso `L1` e `L2` sejam de tamanhos diferentes, completar `L3` com os itens não intercalados da lista de maior número de itens. Seu comportamento é o expresso a seguir:

intercalada

```
?- intercalada([], [], Z).
Z = [] ?
yes

?- intercalada([a], [], Z).
Z = [a] ?
yes

?- intercalada([], [1,2], Z).
Z = [1, 2] ?
yes

?- intercalada([], [1,2,3,4], Z).
Z = [1, 2, 3, 4] ?
yes

?- intercalada([a,b], [1,2,3,4], Z).
Z = [a, 1, b, 2, 3, 4] ?
yes

?- intercalada([a,b,c,d], [1,2,3,4], Z).
Z = [a, 1, b, 2, c, 3, d, 4] ?
yes
```

Exercício 3

Construir um predicado em Prolog, denominado "`insercao_ord`", o qual relaciona um item (I), uma lista ordenada (`L1` – possivelmente vazia) de itens da mesma natureza de I, e uma lista ordenada (`L2` – não vazia). A lista `L2` contem todos os itens de `L1` e ainda o item I em uma posição que mantenha `L2` de forma ordenada. Seu comportamento é o expresso abaixo:

insercao_ord

```
?- insercao_ord( 6, [], Z).
Z = [6] ?
yes

?- insercao_ord( 6, [5], Z).
Z = [5, 6] ?
```

```
yes

?- insercao_ord( 6, [3 ,4 , 5 , 8 , 9, 10], Z).
Z = [3, 4, 5, 6, 8, 9, 10] ?
yes
```

Exercício 4

Construir um predicado em Prolog, denominado “ordenada”, o qual relaciona duas listas (`L1` e `L2` – possivelmente vazias) de itens numéricos, onde `L2` tem exatamente os mesmos itens de `L1`, porém de maneira ordenada. Seu comportamento é o expresso abaixo:

ordenada

```
?- ordenada([ ], Z).
Z = [] ?
yes

?- ordenada([11 , 10, 9], Z).
Z = [9, 10, 11] ?
yes

?- ordenada([11 , 10, 9, 5, 7, 2, 4, 8, 9, 1111, 2, 3, 45, 7,
    888, 989], Z).
 Z = [2, 2, 3, 4, 5, 7, 7, 8, 9, 9, 10, 11, 45, 888, 989,
    1111] ?
yes

?- ordenada([ 5 ], [ 4 ]).
no

?- ordenada([ 5, 6, 7 ], [ 4, 5, 6 ]).
no
```

Exercício 5

Suponha que existam 10 (dez) pessoas em uma cidade, e seus nomes são: “a”, “b”, “c”, “d”, “e”, “f”, “g”, “h”, “i”, “j”. Também é conhecido o fato de que os seguintes pares de pessoas se comunicam freqüentemente:

```
[[a f] [f e] [g i] [h b] [c h] [j d] [g j] [b h] [d i]]
```

Claramente, “a”, “e” e “f” formam uma sub-cultura na qual seus membros se comunicam mutuamente, porém não se comunicam com nenhuma outra sub-cultura da cidade. Quantas sub-culturas existem no conjunto acima (ou em qualquer conjunto semelhante ao conjunto dado) ?

Para ajudar a responder à pergunta acima, construir um predicado em Prolog, denominado "`subcultura`", o qual relaciona duas listas (`L1` e `L2` – possivelmente vazias), as quais englobam sub-listas de apenas 1 (um) nível de profundidade. Cada sub-lista de `L1` é composta de um par de itens (nomes de pessoas) ao passo que cada sub-lista de `L2` representa um grupo com todas as pessoas de uma sub-cultura. Seu comportamento é o expresso abaixo:

subcultura

```
?- subcultura([[a, b], [b, c], [d, e], [e, f]], S).
S = [[a, b, c], [d, e, f]] ?
yes

?- subcultura([[a, e], [b, f], [c, g], [d, g]], S).
S = [[a, e], [b, f], [c, g, d]] ?
yes

?- subcultura([[a, f], [f, e], [g, i], [h, b], [c, h], [j, d
   ], [g, j], [b, h], [d, i]], S).
S = [[b, c, h], [d, g, i, j], [a, e, f]] ?
yes
```

Informações Gerais

- **Nota**: 100 (35%)
- **Data de entrega**: 28/11/18
- **Número de integrantes**: individual

Os algoritmos devem ser implementados utilizando a linguagem de programação Prolog. O código deverá estar disponível no repositório em:

- **inteligencia-artificial/trabalho03/trabalho.pl**

Entrega

A entrega deverá ser realizada via GitHub. Deverá ser criado um repositório chamado **inteligencia-artificial**. O repositório deverá conter a estrutura de diretórios disponibilizada no arquivo **template-repositorio.tgz**. Para oficializar a entrega, o professor deverá ser adicionado como colaborador ao projeto. Para isso vá em **Settings > Collaborators** e adicione o usuário **chaua**. Sem isto o trabalho não será aceito como entregue.

Referências

Exercícios adaptados da apostila de taplog do prof. Alexandre Direne [1].

[1] `http://www.inf.ufpr.br/alexd/tap/taplog.pdf`