

Inteligência Artificial 04/05 — Aula 3

1. Defina um predicado `mylength(L,N)` que sucede se `L` for uma lista e `N` for o seu tamanho. Note que já existe um predicado `length/2` pré-definido no Yap que pode usar para testar o seu predicado. Exemplos de utilização:

```
?- mylength([a,b,c],N).
```

```
N = 3 ? ;
```

```
no
```

```
?- mylength([x,1],2).
```

```
yes
```

```
?- mylength([1,2,3],2).
```

```
no
```

2. Defina um predicado `reverse(L1,L2)` que sucede se `L1` for uma lista e `L2` for igual à lista `L1` invertida. Se necessário, use predicados auxiliares. Exemplos de utilização:

```
?- reverse([a,b,c,d],L).
```

```
L = [d,c,b,a] ? ;
```

```
no
```

```
?- reverse([1],[1]).
```

```
yes
```

3. Investigue o comportamento do predicado que criou na alínea anterior usando as ferramentas de debugging do Yap.

```
?- spy reverse.
```

```
(...)
```

```
?- reverse([a,b,c],L).
```

```
(...)
```

Use o manual do Yap para perceber os comandos que pode usar (`debug`, `nodebug`, `spy`, `nospyspy`, `nospyspyall`, `trace`, `notrace`, `debugging`, etc) e durante o debugging faça `h` para obter um help dos sub-comandos disponíveis.

O que conclui quanto à eficiência do seu predicado `reverse/2`? Consegue criar uma versão melhorada?

4. Considerando o predicado `reverse/2` que fez, o que sucede se fizer a seguinte pergunta no Prolog e for pedindo mais resultados?

```
?- reverse(L,[a,b,c]).
```

Seria válido esperar outro tipo de comportamento do seu predicado neste caso? Torna-se importante poder especificar ao dar a definição do predicado quais são os argumentos que devem ser de input (e portanto devem estar já instanciados), quais são os de output, etc. Em Prolog a notação usual é a seguinte:

```
+Arg -> Deve estar instanciado na chamada ao predicado (input)
-Arg -> Não deve estar instanciado na chamada (output)
?Arg -> Pode ou não estar instanciado
```

Exemplificando, o que lhe foi pedido era: `reverse(+L1,?L2)`.

5. Defina um predicado `separate(+X,+L,?Lsmaller,?Lbigger)` que sucede se `X` for um número, `L` uma lista de números, `Lsmaller` for a lista dos elementos de `L` que são menores ou iguais a `X` e `Lbigger` a lista dos elementos de `L` que são maiores do que `X`. Exemplo de utilização:

```
?- separate(4,[1,9,2,8,7,4,3],L1,L2).
L1 = [1,2,4,3]
L2 = [9,8,7] ? ;
no
```

6. Usando o predicado `separate/2`, crie um predicado `quicksort(+L1,-L2)` que sucede se `L1` for uma lista de números e `L2` for a lista `L1` ordenada por ordem crescente. Tal como o nome indica, para ordenar deve usar o algoritmo quicksort. Exemplo de utilização:

```
?- quicksort([9,1,5,8,2],L).
L = [1,2,5,8,9] ? ;
no
```

7. Consulte as instruções dadas na página das práticas para usar o sistema de avaliação automática Mooshak. Os problemas relativos a esta aula estão disponíveis em `''Aula 3 - Listas''`. Use o Mooshak para descobrir se a sua implementação do `quicksort/2` funciona e aproveite para tentar fazer todos os problemas que estão lá propostos.