Inteligência Artificial — Aula 5

1. Defina o predicado `p(L)` que sucede se `L` fôr uma lista não vazia em que os elementos de ordem ímpar são inteiros maiores ou iguais a zero e os elementos de ordem par são listas com tantos elementos quantos o valor do inteiro na posição precendente.

2. Defina o predicado `elimina_repetidos(L1,L2)` que sucede se `L2` tem os elementos de `L1` pela mesma ordem mas sem repetições.

3. Defina o predicado `palindromo(L)` que sucede se `L` for uma lista não vazia que se lê da mesma forma da esquerda para a direita e da direita para a esquerda.

4. Defina o predicado `troca(L1,L2)` que sucede se `L2` fôr uma lista não vazia e com um número de elementos par que resulta de trocar os elementos na posição ímpar n com os elementos na posição par $n+1$ da lista `L1`. Se os elementos a trocar da lista `L1` forem iguais são removidos da lista `L2`.

5. Defina o predicado `remove_dois(N,M,L1,L2)` que sucede se `L2` fôr uma lista que resulta de remover os elementos na posição `N` e na posição `M` da lista `L1`.

6. Defina o predicado `metades(L1,L2,L3)` que sucede se `L1` fôr uma lista não vazia com um número par de elementos, `L2` fôr uma lista com a primeira metade de `L1` e `L3` fôr uma lista com a segunda metade de `L1`.

7. Defina o predicado `parte_por(S,L1,L2,L3)` que sucede se `L1` fôr uma lista não vazia, `S` fôr um elemento único de `L1` e `L2` fôr uma lista com os elementos de `L1` até `S` e `L3` fôr uma lista com os elementos de `L1` depois de `S`.

8. Defina o predicado `pascal(N,L)`, onde `L` é a enésima linha do triângulo de Pascal.

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
...
```

9. Suponha que numa árvore binária cada nó é representado por um termo `no(V,E,D)`, onde `V` é o valor do nó, `E` (respectivamente, `D`) a sub-árvore esquerda (direita). A árvore vazia é representada por `[]`. Escreva os seguintes predicados Prolog:

(a) `subarvore(A,B)` que dadas duas árvores `A` e `B`, sucede se `B` é uma sub-árvore de `A`.

(b) `ord(T)` que dada uma árvore `T` suceda se `T` for uma árvore ordenada de inteiros, isto é, para cada nó o valor é um inteiro e os valores na subárvore esquerda são menores que esse valor e os valores na subárvore direita são maiores que esse valor.

(c) `pertence(X,A)` que dado um inteiro em `X` e uma árvore ordenada em `A` sucede se `X` está em `A`.

(d) `del(X,T1,T2)` para eliminar o elemento `X` da árvore `T1` construindo em `T2` a árvore ordenada sem esse elemento.