Inteligência Artificial — Aula 9

1. (a) Escreva uma DCG que aceita uma linguagem de controlo de um robot. A linguagem descreve `movimentos`. Cada movimento ou é um `passo` ou é uma sequência de passos. Os passos são `esquerda`, `direita`, `cima` ou `baixo`.
Exemplo:

```
?- movimento([esquerda,direita,baixo,direita,cima],[]).

yes

?- movimento([direita,direita,salta],[]).

no
```

(b) Modifique a DCG anterior para indicar as coordenadas do robot após uma sequência de passos, supondo que as coordenadas iniciais do robot são `(0,0)`.

```
?- movimento((X,Y),[esquerda,cima,direita,baixo],[]).

 X = 0, Y = 0?

?- movimento((X,Y),[esquerda,baixo,direita,direita,baixo,
                    baixo,cima],[]).

 X = 1, Y = 2?
```

(c) Considere que o robot pode funcionar agora em duas velocidades, *lenta* e *turbo*, e que quando em *turbo* a distância percorrida por cada movimento é o dobro. A linguagem agora inclui também as mudanças de velocidade, lenta e turbo, e deve terminar sempre com a palavra *para*. Escreva uma DCG de forma a aceitar a nova linguagem e a calcular, com base nas coordenadas, a distância do robot à origem.

```
?- distancia(D,[lenta,lenta,turbo,direita,cima,lenta,
                direita,direita,baixo,turbo,cima,para],[]).

 D = 5.0 ?
```

2. Escreva uma DCG para reconhecer URLs. O resultado deve ser um termo da forma `url(P,D,L)`, em que `P` é o protocolo (`http`, `file` ou `ftp`), `D` é a lista de domínios por ordem inversa de ocorrência e `L` é a lista de nomes dando a localização do ficheiro também por ordem inversa de ocorrência. Suponha que um nome num dominio/localização, é uma sequência de letras minúsculas e de dígitos.

```
?- url(U,"http://www.dcc.fc.up.pt/aulas/ia0203/dcgs",[]).

 U = url(http,[pt,up,fc,dcc,www],[dcgs,ia0203,aulas]).

?- url(U,"http.aulas/ia0203/dcgs",[]).

 no.
```

3. (a) Escreva uma DCG para reconhecer a seguinte gramática:

$$\texttt{C} := \texttt{C};\texttt{C} \mid \texttt{V} := \texttt{E} \mid \texttt{if B then C else C} \mid \texttt{while B do C}$$
$$\texttt{B} := \texttt{true} \mid \texttt{false} \mid \texttt{B and B} \mid \texttt{B or B}$$
$$\texttt{E} := \texttt{A} \mid \texttt{B}$$

onde `V` é uma sequência de letras ou dígitos que começa com letra minúsculas, e `A` é a seguinte gramática para reconhecer expressões aritméticas:

$$\texttt{A} := \texttt{A} + \texttt{T} \mid \texttt{A} - \texttt{T} \mid \texttt{T}$$
$$\texttt{T} := \texttt{T} * \texttt{F} \mid \texttt{T}/\texttt{F} \mid \texttt{F}$$
$$\texttt{F} := \texttt{number}|\texttt{V}|(\texttt{A}$$

```
?- parse([if,true,then,x1,:,=,23,+,56,else,x2,:,=,false,;,
          x2,:,=,true],[]).
 yes.

?- parse([while,true,do,true],[]).
 no.
```

(b) Modifique a DCG anterior para retornar a árvore sintática do programa.

```
?- parse(A,[if,true,then,x1,:,=,23,+,56,else,x2,:,=,false,;,
          x2,:,=,true],[]).

A = [if(bool(true),atrib(var(x1),arit(op(+),23,56)),
     atrib(var(x2),bool(false))), atrib(var(x2),bool(true))].

?- parse(A,[while,true,do,true],[]).

 no.
```