Inteligência Artificial — Soluções – Aula 9

1. (a)
```
movimento --> passo, movimento.
movimento --> [].

passo --> [direita].
passo --> [esquerda].
passo --> [cima].
passo --> [baixo].
```

(b)
```
movimento(X,Y) --> passo(X1,Y1), movimento(X2,Y2),
                   {X is X1+X2, Y is Y1 + Y2}.
movimento(0,0) --> [].


passo(1,0) --> [direita].
passo(-1,0) --> [esquerda].

passo(0,1) --> [cima].
passo(0,-1) --> [baixo].
```

(c)
```
distancia(D) --> movimento(X,Y,1), {D is sqrt(X*X + Y*Y)}.

movimento(X,Y,_) --> velocidade(V), movimento(X,Y,V).
movimento(X,Y,V) --> passo(X1,Y1), movimento(X2,Y2,V),
                     {X is X1*V + X2, Y is Y1*V + Y2}.
movimento(0,0) --> [para].

passo(1,0) --> [direita].
passo(-1,0) --> [esquerda].
passo(0,1) --> [cima].
passo(0,-1) --> [baixo].

velocidade(1) --> [lenta].
velocidade(2) --> [rapida].
```

2.
```
url(url(P,RD,RL)) --> protocolo(P), dominio(D), local(L),
        {reverse(D,RD), reverse(L,RL)}.

protocolo(P) --> palavra(P),"://", {prot(P)}.
```

```
dominio([P])   --> palavra(P).
dominio([P|R]) --> palavra(P), ".", dominio(R).

local([])    --> [].
local([])    --> "/". % Pode ter barra no fim...
local([P|R]) --> "/", palavra(P), local(R).

palavra(P) --> pal_aux(L),{name(P,L)}.
pal_aux([C])   --> letra(C).
pal_aux([C|T]) --> letra(C), pal_aux(T).

letra(C) --> [C], {C>="a", C=<"z"}.
letra(C) --> [C], {C>="0", C=<"9"}.

prot(http).
prot(file).
prot(ftp).
```

3. (a)

```
parse --> comando, [;], parse.
parse --> comando.

comando --> variavel, [:,=], expr.
comando --> [if], exprb,  [then], comando, [else], comando.
comando --> [while], exprb,  [do], comando.

expr --> expra.
expr --> exprb.


exprb --> bool, [and], exprb.
exprb --> bool, [or], exprb.
exprb --> bool.

bool --> [true].
bool --> [false].


expra --> termo, [+], expra.
expra --> termo, [-], expra.
expra --> termo.

termo --> factor.
termo --> factor, [*], termo.
```

```
termo --> factor, [/], termo.

factor --> [N], {number(N)}.
factor --> variavel.
factor --> ['('], expra, [')'].

variavel --> [N], {atom(N), not token(N)}.

token(if).
token(then).
token(else).
token(while).
token(true).
token(false).
```

(b)
```
parse([C|RC]) --> comando(C), [;], parse(RC).
parse([C]) --> comando(C).

comando(atrib(V,E)) --> variavel(V), [:,=], expr(E).
comando(if(B,C1,C2)) --> [if], exprb(B), [then], comando(C1),
                         [else], comando(C2).
comando(while(B,C)) --> [while], exprb(B),  [do], comando(C).

expr(E) --> expra(E).
expr(E) --> exprb(E).


exprb(and(B,EB)) --> bool(B), [and], exprb(EB).
exprb(or(B,EB)) --> bool(B), [or], exprb(EB).
exprb(B) --> bool(B).

bool(bool(true)) --> [true].
bool(bool(false)) --> [false].

expra(arit(op(-),T,A)) --> termo(T), [+], expra(A).
expra(arit(op(-),T,A))--> termo(T), [-], expra(A).
expra(T) --> termo(T).


termo(arit(op(*),F,T))--> factor(F), [*], termo(T).
termo(arit(op(/),F,T))--> factor(F), [/], termo(T).
termo(F) --> factor(F).
```

```
factor(N) --> [N], {number(N)}.
factor(V) --> variavel(V).
factor(E) --> ['('], expra(E), [')'].

variavel(var(V)) --> [V], {atom(V), not token(V)}.

token(if).
token(then).
token(else).
token(while).
token(true).
token(false).
```