AGENTE001 – MUNDO WUMPUS

DISCIPLINA DE LÓGICA MATEMÁTICA

PROFESSOR: DR. RUBEN CARLO BENANTE

GRUPO 01(COOKIE):

* Matheus Vitor Fernandes Moreira, @matheusvfm
* Mateus Nascimento E Silva, @MateusAKE
* Vinícius Marzo De Araújo, @viniciusmarzo752
* Leonardo Nadolny Magalhães, @leonmagalhaes
* José Roberto Lopes Gentile Almeida, @joseph335
* Gildo da Silva Souza Neto, @tremdo157
* José Lucas Bessa De Oliveira, @JoseLucasBessa

INTRODUÇÃO

- O Mundo Wumpus:
  - Objetivo do jogo
  - Obstáculos e Eventos
  - Percepções e Ações

- Lista de Dinâmicos:
  - Fatos que podem ser modificados

```
/*Dinamicos*/
:- dynamic([ouro/1, arrow/1, casa/1, direcao/1, objetivo/1, wumpus/1, casasvisitadas/1, giro/1]).
```

- O init_agent:
  - Usado para adicionar e limpar memória;
  - Adicionar os dinâmicos.

```
%inicialização
init_agent :-
    retractall(ouro(_)),
    assert(ouro(0)),
    retractall(arrow(_)),
    assert(arrow(1)),
    retractall(casa(_)),
    assert(casa([])),
    retractall(direcao(_)),
    assert(direcao([])),
    retractall(objetivo(_)),
    assert(objetivo(explorar)),
    retractall(wumpus(_)),
    assert(wumpus(vivo)),
    retractall(casasvisitadas(_)),
    assert(casasvisitadas([])),
    retractall(giro(_)),
    assert(giro(0)).
```

- O run_agent:
  - O predicado que se comunica com o mundo Wumpus;
  - Retorna uma ação a determinada percepção;
  - Usado tambem para verificar outros predicados (debug);

```
    %loop do agente
run_agent(P, A) :-
    objetivo(Ob),
    write('Objetivo Atual: '),
    writeln(Ob),
    ouro(O),
    write('Ouro: '),
    writeln(O),
    write('Percepcao: '),
    writeln(P),
    percep(P, Px),
    write('Percebi: '),
    writeln(Px),
    localizacao(P),
    casa(Ca),
    write('Casa atual: '),
    writeln(Ca),
    direcao(Dir),
    write('Direcao: '),
    writeln(Dir),
    giro(Gi),
    write('Giro: '),
    writeln(Gi),
    adjacentes(Adj),
    write('Casas adjacentes: '),
    writeln(Adj),
    casasvisitadas(CV),
    write('Casas visitadas: '),
    writeln(CV),
    agente(P, A).
```

- Predicado “agente”:
  - Dividido em duas partes;
  - Define o que o agente deve fazer dependendo de cada percepção

```
    %objetivos do agente

    %1: se encontrar o ouro
agente([_,_,yes,_,_,_,_], grab) :-
    retract(ouro(A)),
    A1 is A + 1,
    assert(ouro(A1)),
    retract(objetivo(_)),
    assert(objetivo(sair)).

    %definicao de objetivo geral
agente(P, A) :-
    objetivo(O),
    percep(P, S),
    acao(O, S, A).
```

OBJETIVOS DO AGENTE:

- Definem a próxima ação do agente
- Caso do wumpus morto
- Retorno de ações dependendo das percepções

PREDICADO “ACAO”

- Relação com percep.

```
    %lista de acoes que se relacioam com o predicato "percep"
acao(sair, qualquercoisa, A):-
    sai(A).

acao(explorar, nada, A):-
    frente(A).

acao(explorar, trombada, A):-
    trombei(A).

acao(explorar, fedor, A):-
    fedeu(A).

acao(explorar, vento, A):-
    ventou(A).

acao(explorar, ruido, A):-
    ruiu(A).

acao(explorar, trombada, A):-
    viraesquerda(A).
```

PREDICADO “TROMBEI”

- 1° condição: viraesquerda.

- 2° condição: frente.

```
    %condições caso ocorra trombada no mapa
trombei(A):-
    giro(G),
    G < 2,
    G1 is G+1,
    retract(giro(_)),
    assert(giro(G1)),
    viraesquerda(A).

trombei(A):-
    giro(G),
    G =:= 2,
    retract(giro(_)),
    assert(giro(0)),
    frente(A).
```

PREDICADOS: “FEDEU, VENTOU E RUIU”

- Os 3 predicados vão operar de maneira semelhante
- se diferenciando apenas pela percepção a qual eles reagem

PREDICADO “FEDEU”

```
%condições caso sinta o fedor do wumpus
343 fedeu(A):-
344     giro(G),
345     G< 2,
346     G1 is G+1,
347     retract(giro(_)),
348     assert(giro(G1)),
349     viraesquerda(A).
350
351 fedeu(A):-
352     giro(G),
353     G=:= 2,
354     retract(giro(_)),
355     assert(giro(0)),
356     frente(A).
```

- Ao sentir o fedor do wumpus em uma das casas o agente deverá girar para seguir em outra direção

PREDICADO “VENTOU”

```
359    %condições caso sinta a brisa do  buraco
360 ventou(A):-
361    giro(G),
362    G< 2,
363    G1 is G+1,
364    retract(giro(_)),
365    assert(giro(G1)),
366    viraesquerda(A).
367
368 ventou(A):-
369    giro(G),
370    G=:= 2,
371    retract(giro(_)),
372    assert(giro(0)),
373    frente(A).
```

- Ao sentir a brisa do buraco em uma das casas o agente deverá girar para seguir em outra direção

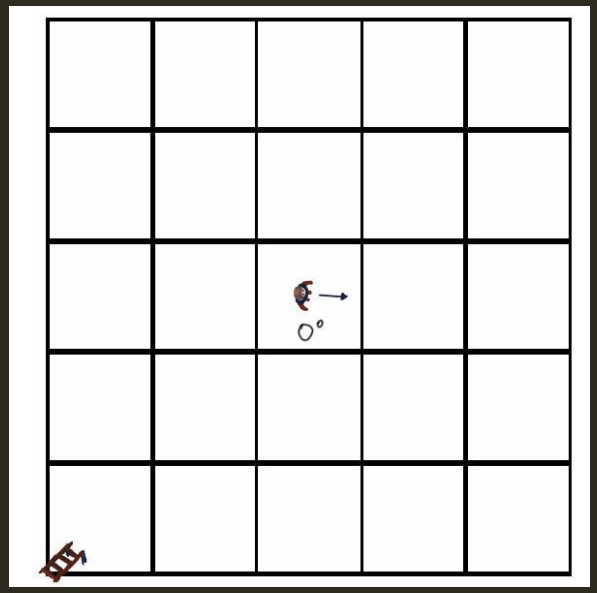
PREDICADO “RUIU”

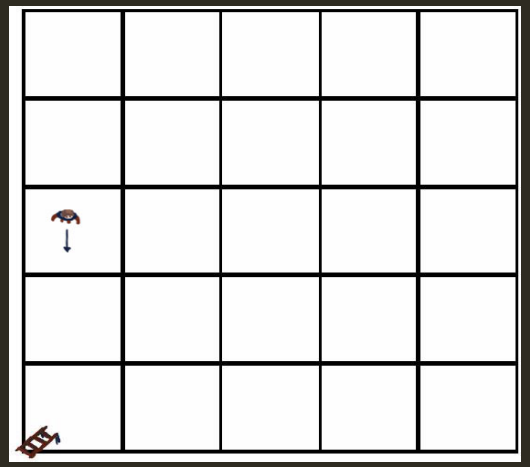
```
%condições caso sinta as farfalhadas do morcego
378 ruiu(A):-
379     giro(G),
380     G< 2,
381     G1 is G+1,
382     retract(giro(_)),
383     assert(giro(G1)),
384     viraesquerda(A).
385
386 ruiu(A):-
387     giro(G),
388     G=:= 2,
389     retract(giro(_)),
390     assert(giro(0)),
391     frente(A).
```

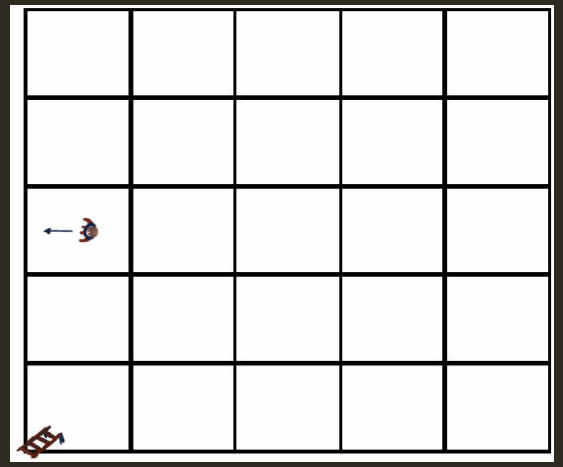
- Ao sentir a as farfalhadas dos morcegos em uma das casas o agente deverá girar para seguir em outra direção

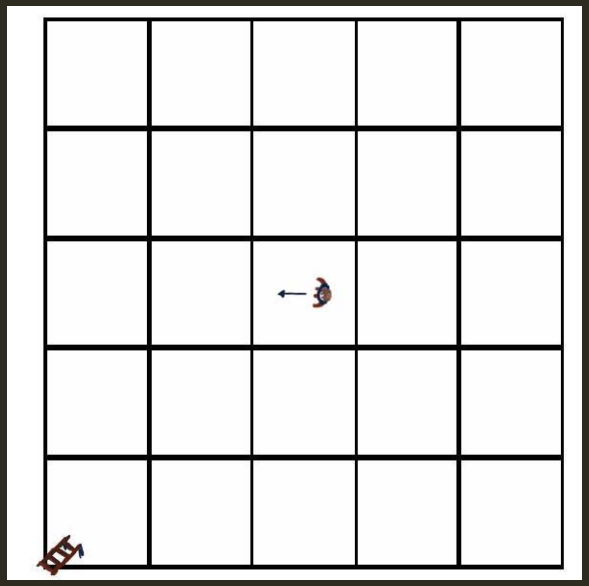
PREDICADO “SAI”:

- Condições para o predicado sair do wumpus
- Muda a casa de saída para a inicial e retorna a ação climb

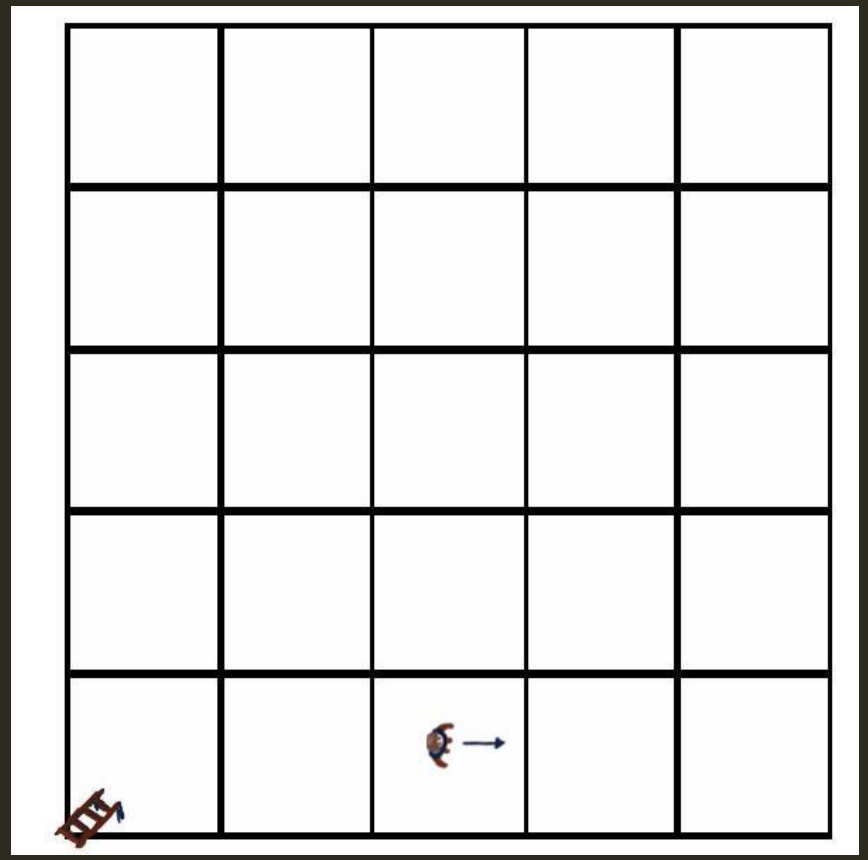








PREDICADO: FRENTE

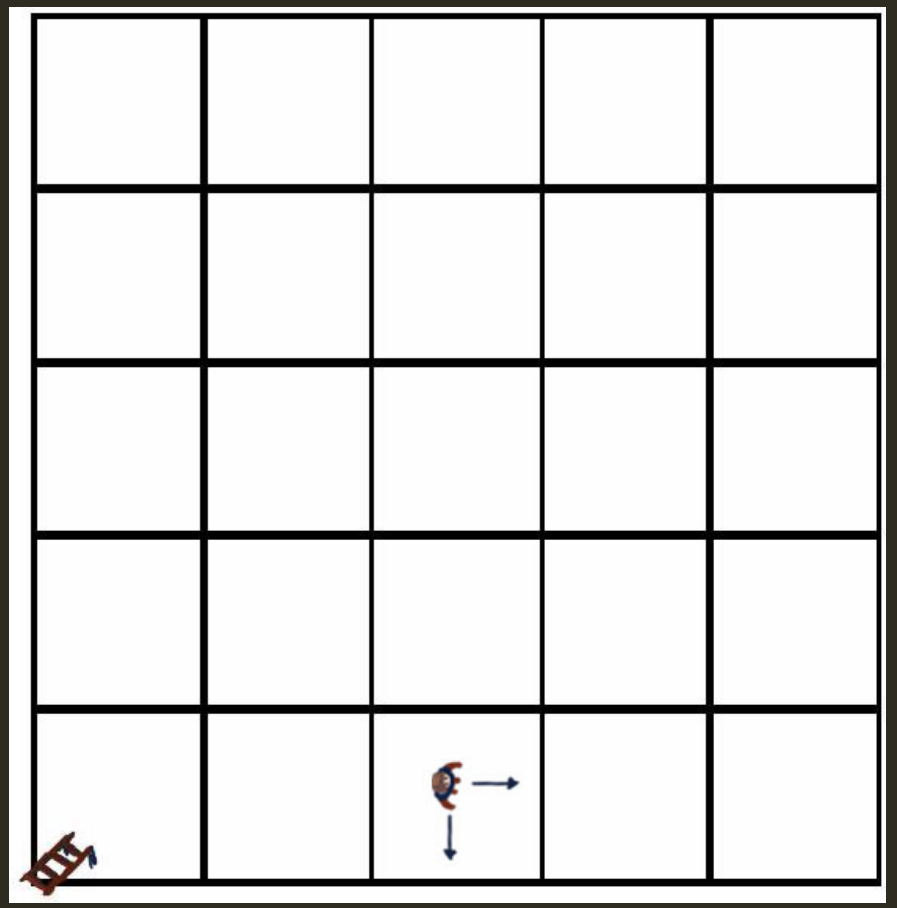


```
      %caso o agente não tenha outra opcao, condições para andar pra frente
frente(goforward) :-
    direcao([0]),
    casa([X,Y]),
    casasvisitadas(CV),
    append(CV, [[X,Y]], L),
    retract(casasvisitadas(_)),
    Xf is X+1,
    retract(casa(_)),
    assert(casasvisitadas(L)),
    assert(casa([Xf, Y])).
```

PREDICADO “VIRAESQUERDA":

```
    %caso caia na trombada, predicato para virar a esquerda
viraesquerda(turnleft):-
    direcao([270]),
    retract(direcao(_)),
    assert(direcao([0])).

viraesquerda(turnleft):-
    direcao([A]),
    A =\= 270,
    Af is A+90,
    retract(direcao(_)),
    assert(direcao([Af])).
```



PREDICADO ADJACENTES:

Predicado Auxiliar que entrega uma lista de casas adjacentes

```
    assert(direcao(Nov)).

localizacao([_,_,_,_,_,_,_,_]). %Caso nao seja a primeira casa ou nao pedir gps


    %lista as casas adjacentes ao agente
adjacentes([C,B,E,D]) :-
    casa(P),
    cima(P, C),
    baixo(P, B),
    esquerda(P, E),
    direita(P, D).

cima([X,Y], [X,Yf]) :-
    Yf is Y+1.

baixo([X,Y], [X,Yf]) :-
    Yf is Y-1.

esquerda([X,Y], [Xf, Y]) :-
    Xf is X-1.

direita([X,Y], [Xf, Y]) :-
    Xf is X+1.
```