Universidade Federal de Pelotas

Conceitos de Linguagens de Programação

Relatório do Projeto
Prolog: Movimentando um Robô

Alunos Renata Junges
Yan Soares

Professor Gerson Cavalheiro

Pelotas, 4 de Dezembro de 2015

Conteúdo

1. Introdução

Esse relatório tem o objetivo de mostrar uma aplicação de um código na linguagem Prolog, que é uma linguagem de uso geral, especialmente associada com a inteligência artificial e linguística computacional.

O problema proposto envolve o deslocamento de um robô em um plano cartesiano de dimensões 100x100, como mostrado na seção 2. O robô pode se deslocar em todos os sentidos – norte, sul, leste, oeste – e pode andar quantos passos quiser, desde que não encontre um obstáculo – parede preta – e não exceda os limites do plano cartesiano.

1.1 Como executar o programa

O programa é executado da seguinte maneira: abre-se o terminal de controle, se muda o diretório até a pasta que o código-fonte está incluso, digita-se **swipl** para abrir a interface do compilador SWI para Prolog e, logo após, **consult(roboprolog).** para que o cenário e as regras sejam inicializados. Após isso já é possível digitar os comandos que movimentarão o robô.

Exemplos de comandos que podem ser digitados:

movimento([virar,para,esquerda]).
movimento([andar,4,passos,para,a,direita]).
movimento([andar,dez,passos,para, o, leste]).
movimento([virar,norte]).
movimento([andar,34,passos, em, frente]).

Todos os comandos mencionados acima podem ser usados nesse programa, o primeiro comando exposto só muda o sentido que o o robô está virado; o segundo comando faz o robô se locomover 4 posições para a direita caso não exista obstáculos. O último comando mostrado, por exemplo, faz com que o robô ande 34 passos na direção que ele está virado, ou seja, em frente. Além disso, números de 0 a 20 podem ser expressos de forma sentencial – zero, … , vinte – e todos os números podem ser expressos na sua forma algébrica também. Quando o programa for inicializado o robô estará na posição (2,2) virado para o norte.

2. Código em Prolog

O código-fonte a seguir deve ser salvo com o nome de **roboprolog.pl** para que funcione de acordo com o tutorial da seção 1.1. Além disso, o programa não contém nenhum bug e as linhas verdes – possuem o símbolo % – caracterizam comentários no código.

2.1 Código-fonte

```
% O robô deve ser declarado como dinâmico para poder atualizar suas posições após
s movimento %
:- dynamic robo/3.

eixoX(100).    % Dimensão horizontal da matriz %
eixoY(100).    % Dimensão vertical da matriz %
robo(2,2,norte). % Robô inicializa no Ponto(2,2), virado para a posição norte %

% Definição do plano: Cores das paredes - Preto é obstáculo, branco é livre %

 % Essas 4 primeiras linhas criam as paredes pretas interiores ao plano %
celula(1,_,preto).
celula(Dim,_,preto) :- eixoX(Dim).
celula(_,1,preto).
celula(_,Dim,preto) :- eixoY(Dim).
 % Parede de (50, Y), onde o Y vai de 100 até 50 %
celula(X,Y,preto) :- eixoX(DimX) , eixoY(DimY) , X is DimX/2 , Y >= DimY/2.
% Parede de (25, Y), onde o Y vai de 1 até 50 %
celula(X,Y,preto) :- eixoX(DimX), eixoY(DimY), X is DimX/4, Y =< DimY/2.
% Parede de (75, Y), onde o Y vai de 1 até 50 %
celula(X,Y,preto) :- eixoX(DimX), eixoY(DimY), X is DimX-25, Y =< DimY/2.
% Tudo que não for parede preta, é parede branca(livre) %
celula(X,Y,branco) :- not(celula(X,Y,preto)).

% Para consulta da posição atual do robô %

posicaoatual() :-
  robo(X,Y,Z),
  format('A posição atual do robô é (x,y): (~w,~w) e está virado para o ~w',[X,Y
,Z]).
```

```
% Printa a mensagem de que não é possivel movimentar %

naopodeandar() :-
 format('O robô não pode andar até o ponto desejado pois existe um obstáculo no cam
inho. '),posicaoatual().

% Comando para só virar o lado do robô %

vira(Lado) :-
  (Lado = 'nenhum' -> format('Nenhum lado foi especificado para o movimento');
  retract(robo(X,Y,_)),
  assert(robo(X,Y,Lado)),
  format('Robô virou para sentido ~w',[Lado])).

% Função para movimentar o robô %

calcula(Numero, Lado) :-
  robo(X,Y,_),
  ( Lado = 'oeste' -> somaesquerda(X,Y,Numero,Pode), movimenta(Pode,X,Y,Numero), !;
    Lado = 'leste' -> somadireita(X,Y,Numero,Pode), movimenta(Pode,X,Y,Numero), !;
    Lado = 'norte' -> somacima(X,Y,Numero,Pode), movimenta(Pode,X,Y,Numero), !;
    Lado = 'sul' -> somabaixo(X,Y,Numero,Pode), movimenta(Pode,X,Y,Numero), ! ).

movimenta(Pode,X,Y,Numero) :-
  ( Pode = 'nao' -> naopodeandar(), !;
    Pode = 'esquerda' -> H is X-Numero, retract(robo(_,_,_)), asserta(robo(H,Y,oest
e)), posicaoatual(), !;
    Pode = 'direita' -> H is X+Numero, retract(robo(_,_,_)), asserta(robo(H,Y,leste
)), posicaoatual(), !;
    Pode = 'cima' -> H is Y+Numero, retract(robo(_,_,_)), asserta(robo(X,H,norte)),
 posicaoatual(), !;
    Pode = 'baixo' -> H is Y-Numero, retract(robo(_,_,_)), asserta(robo(X,H,sul)),
posicaoatual(), ! ).
somaesquerda(X,Y,Numero,Pode) :-
  celula(X,Y,Cor),
  ( Cor = preto -> Pode = 'nao', !;
       Numero is 0 -> Pode = 'esquerda', !;
    Cor = branco -> A is X - 1, B is Numero - 1, somaesquerda(A,Y,B,Pode) ).

somacima(X,Y,Numero,Pode) :-
  celula(X,Y,Cor),
  ( Cor = preto -> Pode = 'nao', !;
       Numero is 0 -> Pode = 'cima', !;
    Cor = branco -> A is Y + 1, B is Numero - 1, somacima(X,A,B,Pode) ).

somadireita(X,Y,Numero,Pode) :-
  celula(X,Y,Cor),
  ( Cor = preto -> Pode = 'nao', !;
       Numero is 0 -> Pode = 'direita', !;
    Cor = branco -> A is X + 1, B is Numero - 1, somadireita(A,Y,B,Pode) ).

somabaixo(X,Y,Numero,Pode) :-
  celula(X,Y,Cor),
  ( Cor = preto -> Pode = 'nao', !;
       Numero is 0 -> Pode = 'baixo', !;
    Cor = branco -> A is Y - 1, B is Numero - 1, somabaixo(X,A,B,Pode) ).
```

```
% Reconhecimento da entrada %

movimento([X|R]) :-
  (X = 'virar' -> pegalado(R, Lado), vira(Lado); X = 'andar' -> pegapassos(R, Numer
o), pegalado(R, Lado), calcula(Numero, Lado)).

pegalado([], 'nenhum'). % Se a lista está vazia e não achou nenhum dos lados, não h
á para onde virar %
pegalado([X|R], Lado) :-
  robo(_,_,Sentido),
  (X = 'frente' -> Lado = Sentido;
   X = 'esquerda' -> Lado = 'oeste';
   X = 'direita' -> Lado = 'leste';
   X = 'baixo' -> Lado = 'sul';
   X = 'cima' -> Lado = 'norte';
   X = 'norte' -> Lado = 'norte';
   X = 'sul' -> Lado = 'sul';
   X = 'oeste' -> Lado = 'oeste';
   X = 'leste' -> Lado = 'leste'; pegalado(R,Lado)).


% Se a lista estiver vazia e não achou nenhum número, não há movimento %
pegapassos([], 0).
pegapassos([X|R], Numero) :-
  (integer(X) -> Numero is X;
   X = 'zero' -> Numero is 0;
   X = 'um' -> Numero is 1;
   X = 'dois' -> Numero is 2;
   X = 'tres' -> Numero is 3;
   X = 'quatro' -> Numero is 4;
   X = 'cinco' -> Numero is 5;
   X = 'seis' -> Numero is 6;
   X = 'sete' -> Numero is 7;
   X = 'oito' -> Numero is 8;
   X = 'nove' -> Numero is 9;
   X = 'dez' -> Numero is 10;
   X = 'onze' -> Numero is 11;
   X = 'doze' -> Numero is 12;
   X = 'treze' -> Numero is 13;
   X = 'catorze' -> Numero is 14;
   X = 'quatorze' -> Numero is 14;
   X = 'quinze' -> Numero is 15;
   X = 'dezesseis' -> Numero is 16;
   X = 'dezessete' -> Numero is 17;
   X = 'dezoito' -> Numero is 18;
   X = 'dezenove' -> Numero is 19;
   X = 'vinte' -> Numero is 20; pegapassos(R,Numero)).
```

2.2 Comentários sobre conjuntos de regras

O reconhecimento da entrada do programa é feito através da regra **movimento**, nessa regra vai ser identificado se a primeira palavra passada é “virar” ou “andar”, pois dependendo de qual foi a palavra, o programa seguirá rumos diferentes.

Se a primeira palavra for “virar”, então o programa chama a regra **pegalado**, e descobre para qual lado o robô deve ser virado, e depois disso, chama-se a regra **vira** que executa a ação de virar o robô. Obviamente, o lado para o qual o robô deve ser virado, tem que vir necessariamente na frase de entrada, os lados/direções que serão aceitas são: frente, norte, sul, leste, oeste, direita, esquerda, cima, baixo. **movimento([virar,para,a,direita]).** ou **movimento([virar,norte]).** são exemplos de entrada para que esse programa funcione.

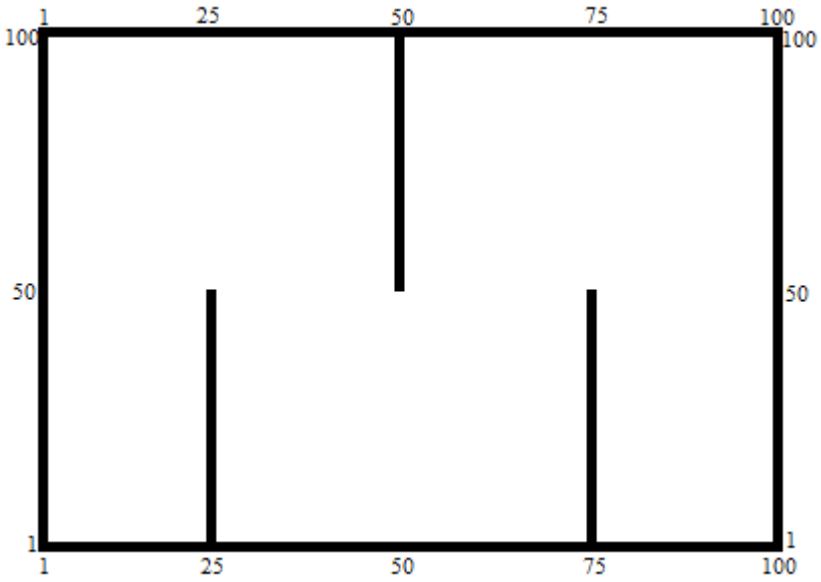
Porém, caso a primeira palavra do movimento seja “andar”, os comandos que serão executados a seguir serão diferentes. Primeiro é necessário saber o número de passos/células que o robô deve andar, para isso é chamada a regra **pegapassos**, depois é necessário saber para qual direção, então – assim como na quando a palavra é “virar” – é chamada a regra **pegalado**; após a obtenção desses 3 atributos é chamada a regra **calcula**.

Finalmente, a regra **calcula**, baseada no número de passos e na direção que se deseja caminhar, chama outras regras que calculam, se é possível ou não, andar na direção proposta sem encontrar obstáculos. Essas regras, são feitas de forma recursiva, sempre decrementando o valor do número de passos (quando for 0 a recursão para) e verificando se não encontrou obstáculo (se encontrar, a recursão para), a variável **Pode**, é uma flag, ela faz o controle se é possível andar e para qual direção foi andada.

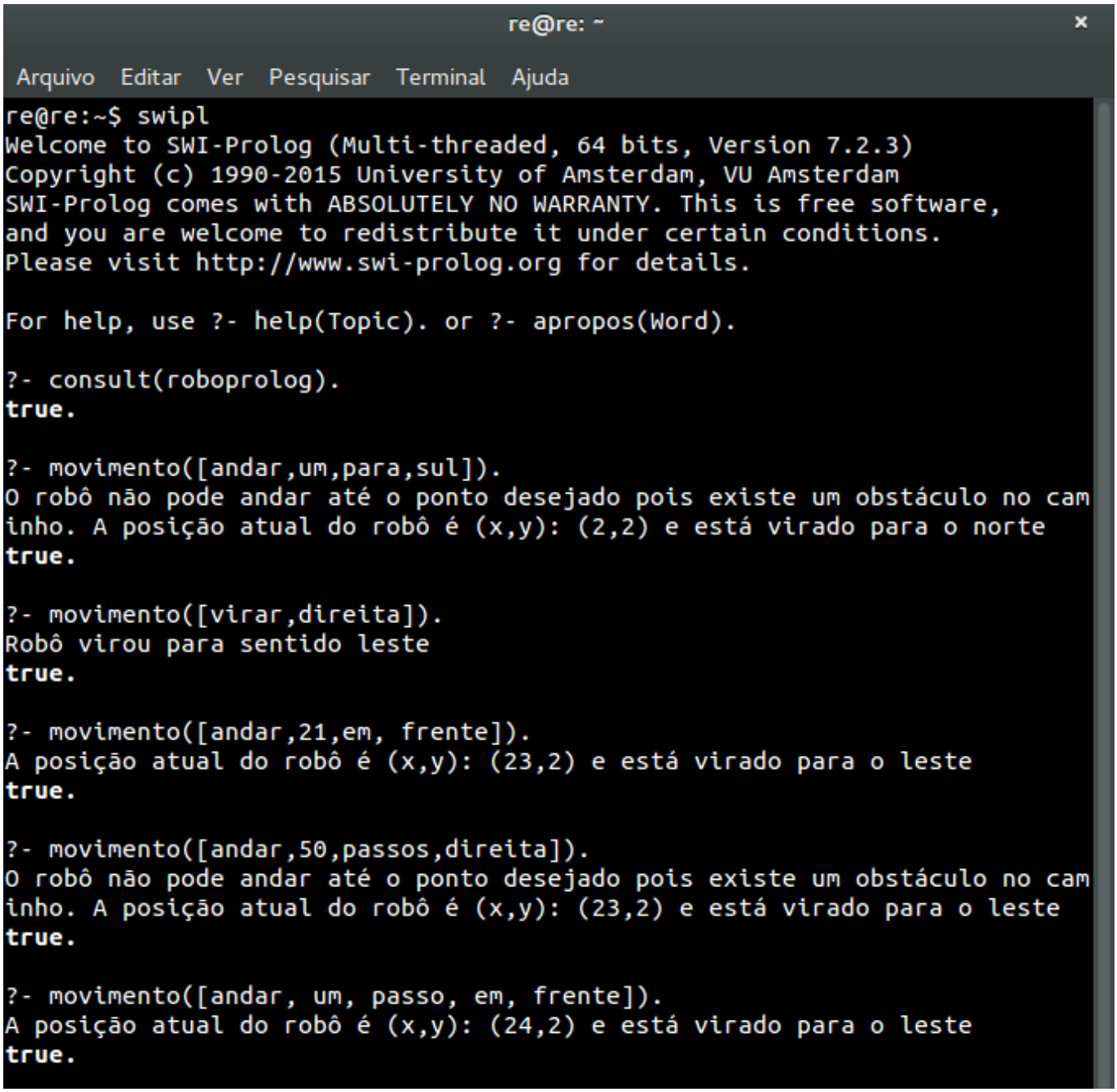
3. Mapa referente ao código

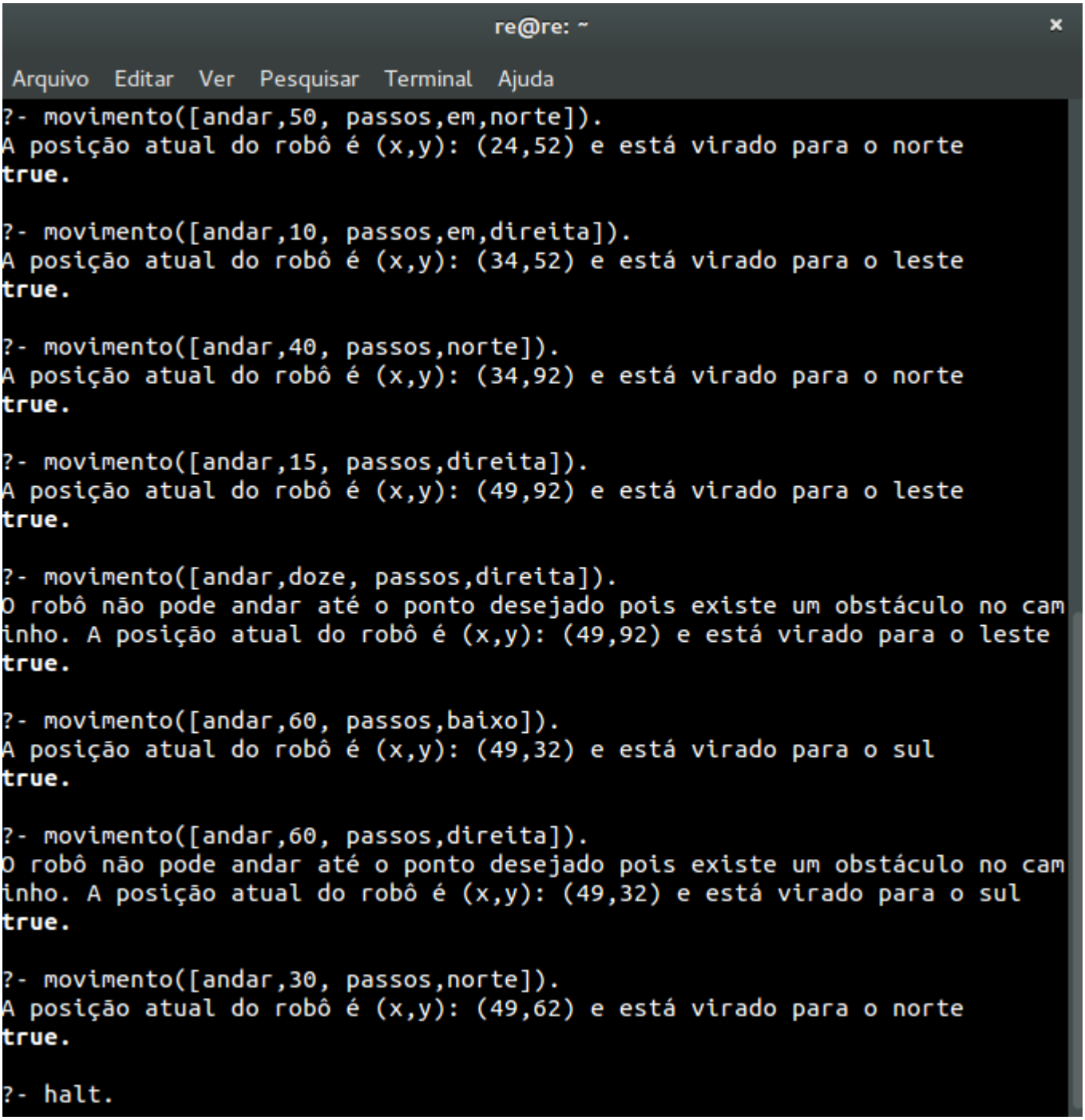
O desenho do mapa que será mostrado a seguir mostra os obstáculos no plano(x , y). Como pode ser visto na imagem, os obstáculos são representados pela linha preta, essa linha está localizada nas bordas do plano (1 <= x <= 100, e 1 <= y <= 100) , e existem três paredes pretas dentro do plano (x = 50 e y >= 50; x = 25 e y <= 50; x = 75 e y <= 50).

Como dito anteriormente, o robô inicializa no ponto (2 , 2) do mapa, virado para norte.



4. Um exemplo de execução




re@re: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
?- movimento([andar,50, passos,em,norte]).
A posição atual do robô é (x,y): (24,52) e está virado para o norte
true.
?- movimento([andar,10, passos,em,direita]).
A posição atual do robô é (x,y): (34,52) e está virado para o leste
true.
?- movimento([andar,40, passos,norte]).
A posição atual do robô é (x,y): (34,92) e está virado para o norte
true.
?- movimento([andar,15, passos,direita]).
A posição atual do robô é (x,y): (49,92) e está virado para o leste
true.
?- movimento([andar,doze, passos,direita]).
O robô não pode andar até o ponto desejado pois existe um obstáculo no cam
inho. A posição atual do robô é (x,y): (49,92) e está virado para o leste
true.
?- movimento([andar,60, passos,baixo]).
A posição atual do robô é (x,y): (49,32) e está virado para o sul
true.
?- movimento([andar,60, passos,direita]).
O robô não pode andar até o ponto desejado pois existe um obstáculo no cam
inho. A posição atual do robô é (x,y): (49,32) e está virado para o sul
true.
?- movimento([andar,30, passos,norte]).
A posição atual do robô é (x,y): (49,62) e está virado para o norte
true.
?- halt.

4.1 Mapa referente à execução