Trabalho 1 - Inteligência Artificial

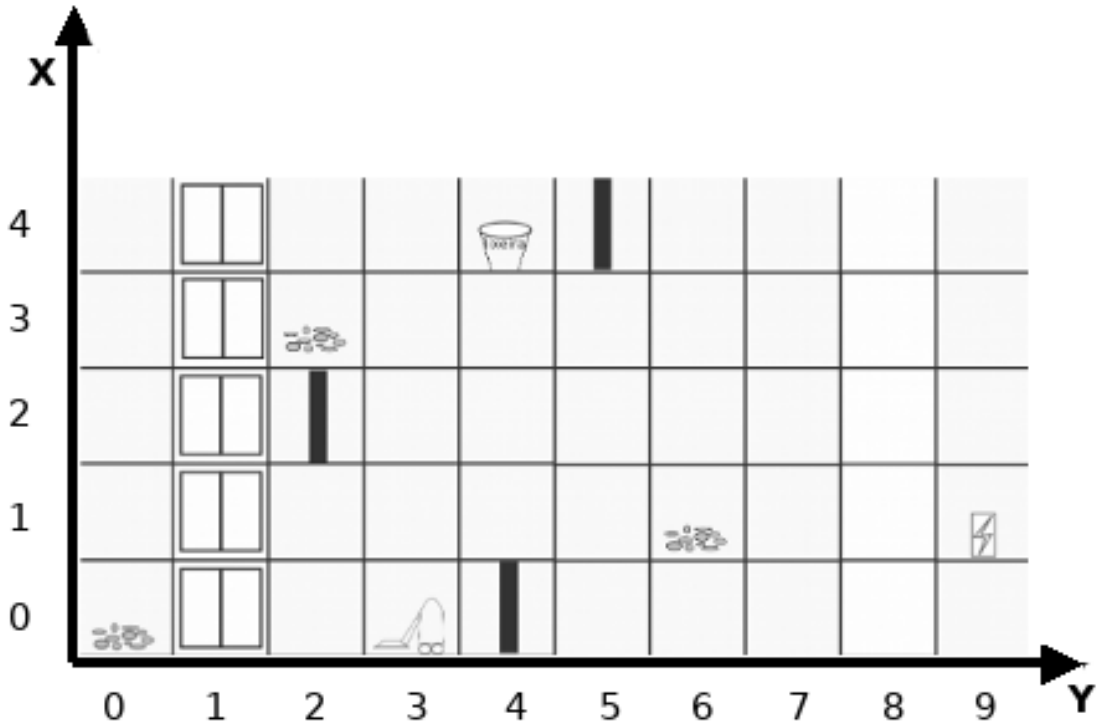
Alisson Hayasi, 726494
Bruna Zamith, 628093

May 17, 2019

Apresentação

1. Modelagem - Estados e Transição

- Estado: Posição atual do AADP (X,Y);
- Transição: Baseada nas regras de movimentação para o AADP (vertical e horizontal);
- Ações tratadas na regra de busca (Simplificação).



Fonte: Baseado no Material de Apoio, professor Dr. Murilo Naldi Coelho.

1. Modelagem - Busca Cega

- Busca em Profundidade;
- Combinação de diferentes buscas:

Cenário	**Meta**
Capacidade máxima não foi atingida, ainda existem sujeiras no cenário	Encontrar sujeira
Capacidade máxima foi atingida, ainda existem sujeiras no cenário	Encontrar lixeira
Não existem sujeiras no cenário	Encontrar *Dock Station*. Encerra.

2. Regras - Ambiente

- Escalável (Genérico);
- Tamanho m*x*n do cenário;
- Capacidade máxima do AADP;
- Lista de sujeiras.

```
cenario(10,5).
parede(4,0).
elevador(1,0).
dockStation(0,0).
lixeira(4,4).
sujeiras([[2,3],[6,1],[6,4]]).
capacidadeMax(2).
```

2. Regras - Gerais

- Manipulação de listas;
- Presentes no corpo de regras futuras;

```
pertence(Entrada1,Entrada2).
concatena(Entrada1,Entrada2,Saida).
pop(Entrada,Saida1,Saida2).
reverse(Entrada,Saida).
```

2. Regras - Movimentação

- ***moveDir***
- ***sobeElev***
- ***moveEsq***
- ***desceElev***

2. Regras - Movimentação

```
% moveDir(Entrada1,Entrada2,Saida1,Saida2)
% Move para direita
% Entrada1: X inicial
% Entrada2: Y inicial
% Saida1: X final
% Saida2: Y final
moveDir(Xin,Y,Xout,Y) :-
    cenario(Lim,_),
    LimDir is Lim -1, % Eixo comeca no 0
    Xin < LimDir,
    X is Xin + 1,
    not(parede(X,Y)),
    Xout is X.
```

2. Regras - Movimentação

```
% sobeElev(Entrada1,Entrada2,Saida1,Saida2)
% Sobe pelo elevador
% Entrada1: X inicial
% Entrada2: Y inicial
% Saida1: X final
% Saida2: Y final
sobeElev(X,Yin,X,Yout) :-
    elevador(X,Yin),
    cenario(_,Lim),
    LimSup is Lim - 1, % Eixo comeca no 0
    Yin < LimSup,
    Yout is Yin + 1.
```

2. Regras - Movimentação

```
% moveEsq(Entrada1,Entrada2,Saida1,Saida2)
% Move para esquerda
% Entrada1: X inicial
% Entrada2: Y inicial
% Saida1: X final
% Saida2: Y final
moveEsq(Xin,Y,Xout,Y) :-
    Xin > 0,
    X is Xin - 1,
    not(parede(X,Y)),
    Xout is X.
```

2. Regras - Movimentação

```
% desceElev(Entrada1,Entrada2,Saida1,Saida2)
% Desce pelo elevador
% Entrada1: X inicial
% Entrada2: Y inicial
% Saida1: X final
% Saida2: Y final
desceElev(X,Yin,X,Yout) :-
    elevador(X,Yin),
    Yin > 0,
    Yout is Yin - 1.
```

2. Regras - Transição

- Regras de transição de estados;
- Baseadas nos movimentos anteriormente vistos;
- Sucessor.

```
% s(Entrada,Saida)
% Transicao de estado
% Entrada: Estado inicial
% Saida: Estado final (sucessor)
s([X,Y],[X1,Y]) :-
    moveDir(X,Y,X1,Y).
s([X,Y],[X,Y1]) :-
    sobeElev(X,Y,X,Y1).
s([X,Y],[X1,Y]) :-
    moveEsq(X,Y,X1,Y).
s([X,Y],[X,Y1]) :-
    desceElev(X,Y,X,Y1).
```

2. Regras - Buscas

- Busca em profundidade: ***meta*** e ***busca***.

```
meta(Estado,Meta) :- Meta == Estado.

busca(Caminho,Estado,Final,[Estado|Caminho]) :-
    meta(Estado,Final).
busca(Caminho,Estado,Final,Solucao) :-
    s(Estado,Sucessor),
    not(pertence(Sucessor,[Estado|Caminho])),
    busca([Estado|Caminho],Sucessor,Final,Solucao).
```

2. Regras - Buscas

- 3 definições para ***limpa***;
- Recebe:
  - Estado atual;
  - Capacidade atual do AADP;
  - Lista de sujeiras no cenário;
  - Caminho atual (ou inicial);
- Retorna: Caminho percorrido até a meta.

2. Regras - Buscas

- Quando não tiver mais sujeira, vai para a *Dock Station*;
- Regra base.

```
limpa(Estado,_,[],Caminho,SolucaoFinal) :-
    dockStation(X,Y),
    busca([],Estado,[X,Y],Solucao),
    concatena(Solucao,Caminho,SolucaoSaida),
    reverse(SolucaoSaida,SolucaoFinal).
```

2. Regras - Buscas

- Busca sujeira se a capacidade atual for menor que capacidade máxima.

```
limpa(Estado,Capacidade,Sujeiras,Caminho,
    CaminhoResposta) :-
        capacidadeMax(CapacidadeMax),
        Capacidade < CapacidadeMax,
        pop(Sujeiras,Alvo,NovaSujeiras),
        busca([],Estado,Alvo,Solucao),
        NovaCapacidade is Capacidade + 1,
        concatena(Solucao,Caminho,NovoCaminho),
        limpa(Alvo,NovaCapacidade,NovaSujeiras,
            NovoCaminho,CaminhoResposta).
```

- Esvazia na lixeira se a capacidade atual for igual à capacidade máxima.

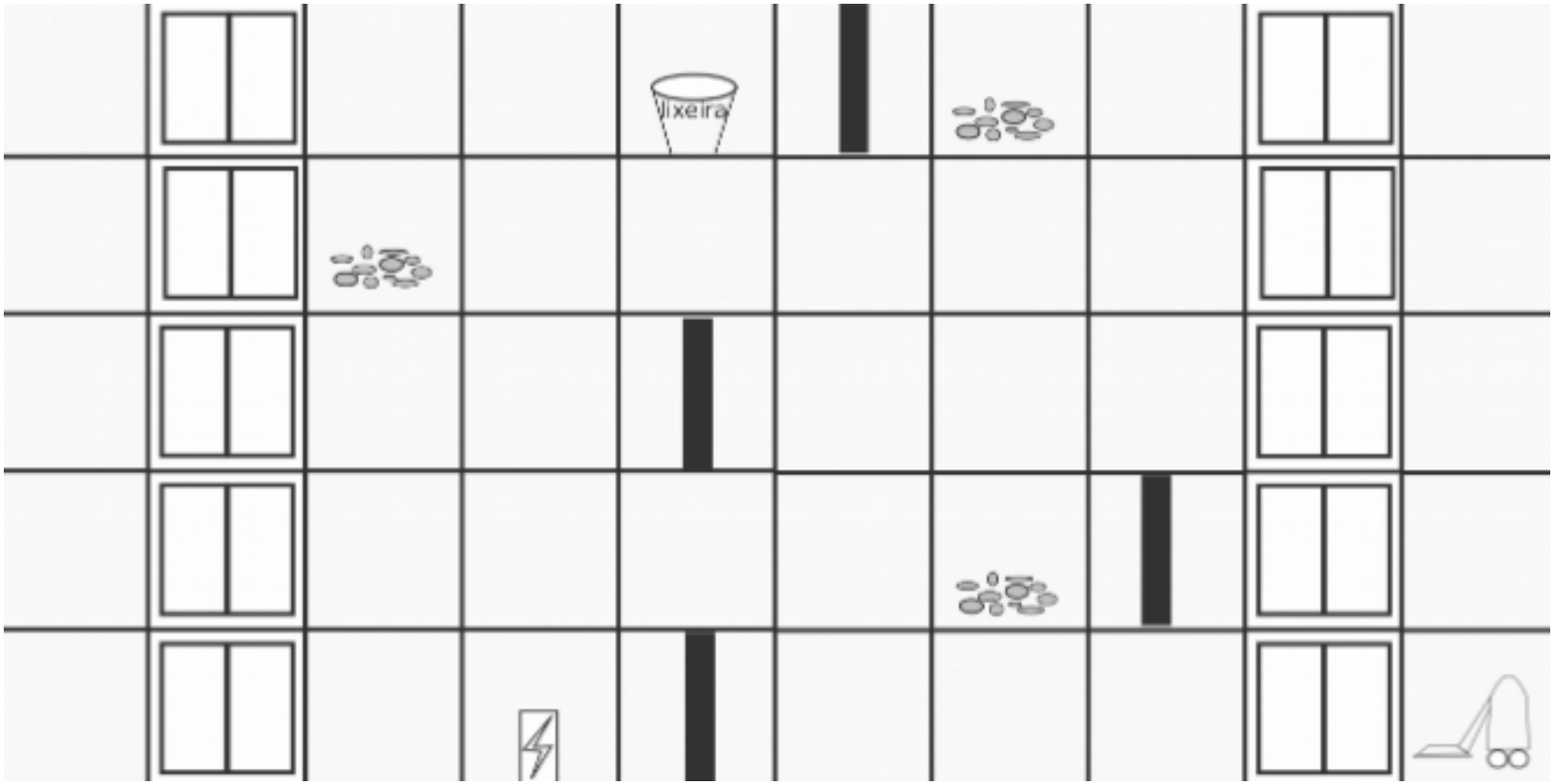
```
limpa(Estado,Capacidade,Sujeiras,Caminho,
    CaminhoResposta) :-
        capacidadeMax(CapacidadeMax),
        Capacidade >= CapacidadeMax,
        lixeira(X,Y),
        busca([],Estado,[X,Y],Solucao),
        concatena(Solucao,Caminho,NovoCaminho),
        limpa([X,Y],0,Sujeiras,NovoCaminho,
            CaminhoResposta).
```

2. Regras - Buscas

- Chamada pelo usuário: Estado Final -> Caminho.

```
solucao(Inicial,Solucao) :- sujeiras(Sujeiras),
    limpa(Inicial,0,Sujeiras,[],Solucao).
```

3. Execução - Cenário 1



Fonte: Baseado no Material de Apoio, professor Dr. Murilo Naldi Coelho.

3. Execução - Cenário 1

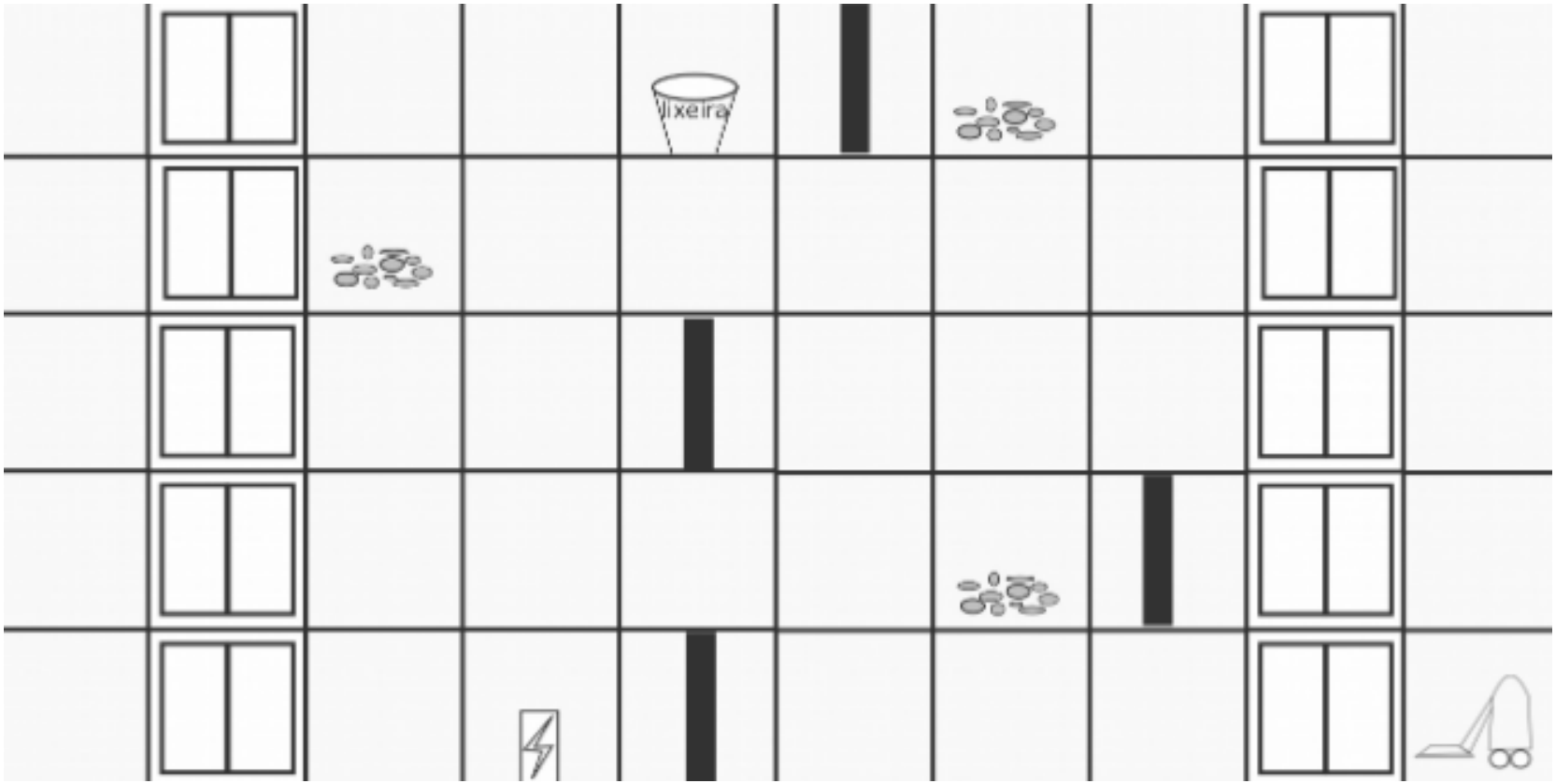
```
% Tamanho Cenario
cenario(10,5).
% Paredes
parede(4,0).
parede(7,1).
parede(4,2).
parede(5,4).
% Elevadores
elevador(1,0).
elevador(1,1).
elevador(1,2).
elevador(1,3).
elevador(1,4).
elevador(8,0).
elevador(8,1).
elevador(8,2).
elevador(8,3).
elevador(8,4).
```

3. Execução - Cenário 1

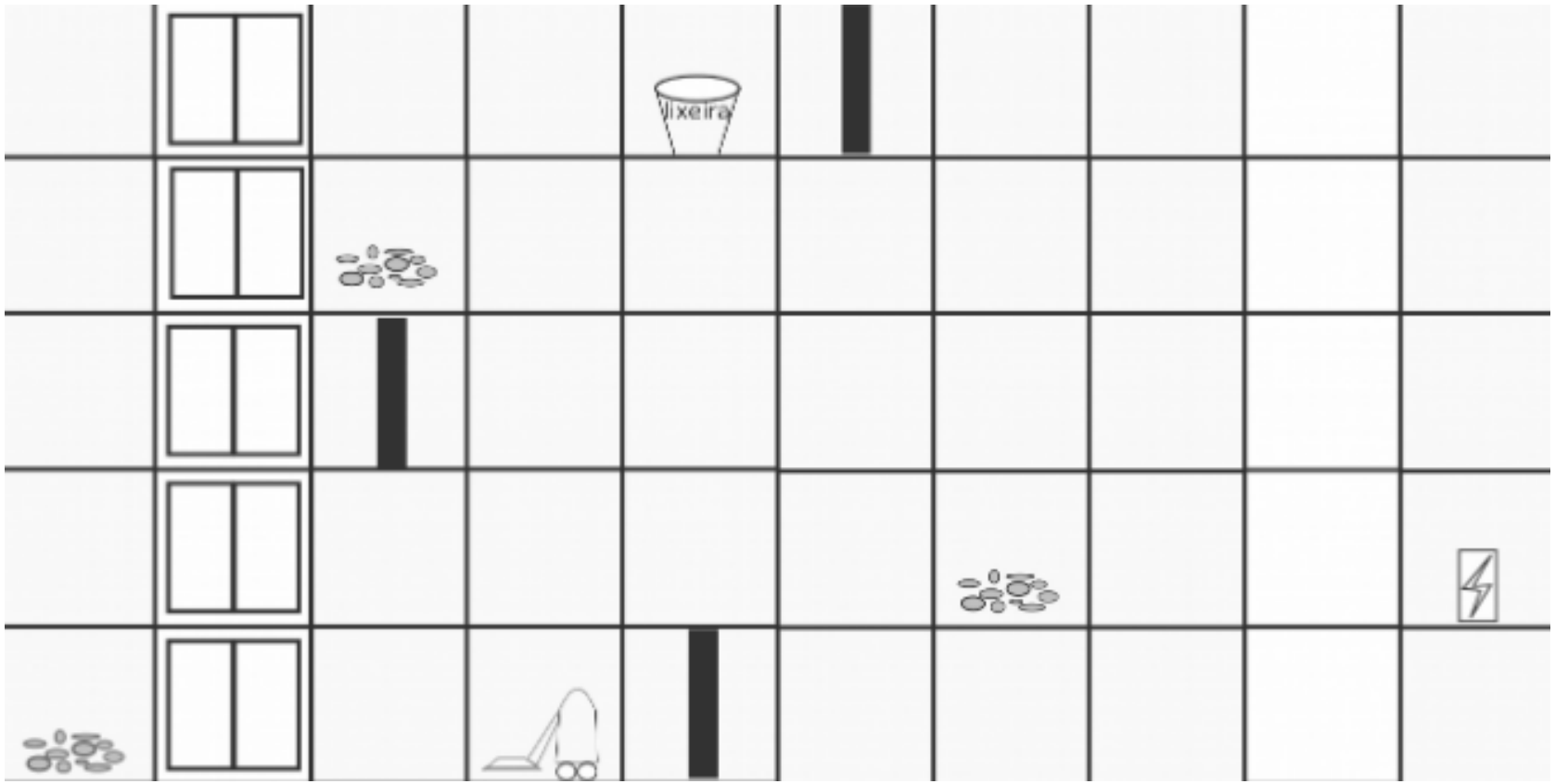
```
% Dock Station
dockStation(3,0).
% Lixeira
lixeira(4,4).
% Sujeiras
sujeiras([[2,3],[6,1],[6,4]]).
% Capacidade AADP
capacidadeMax(2).
```

3. Execução - Cenário 1

```
?- solucao([9,0],Solucao).
Solucao = [[9, 0], [8, 0], [8, 1], [8, 2], [8, 3], [7, 3], [6, 3], [5, 3], [4, 3], [3, 3], [2, 3], [2, 3]
, [1, 3], [1, 2], [1, 1], [2, 1], [3, 1], [4, 1], [5, 1], [6, 1], [6, 1], [5, 1], [4, 1], [3, 1], [2, 1],
 [1, 1], [1, 2], [1, 3], [1, 4], [2, 4], [3, 4], [4, 4], [4, 4], [3, 4], [2, 4], [1, 4], [1, 3], [2, 3],
[3, 3], [4, 3], [5, 3], [6, 3], [7, 3], [8, 3], [8, 4], [7, 4], [6, 4], [6, 4], [7, 4], [8, 4], [8, 3], [
7, 3], [6, 3], [5, 3], [4, 3], [3, 3], [2, 3], [1, 3], [1, 2], [1, 1], [1, 0], [2, 0], [3, 0]] .
```



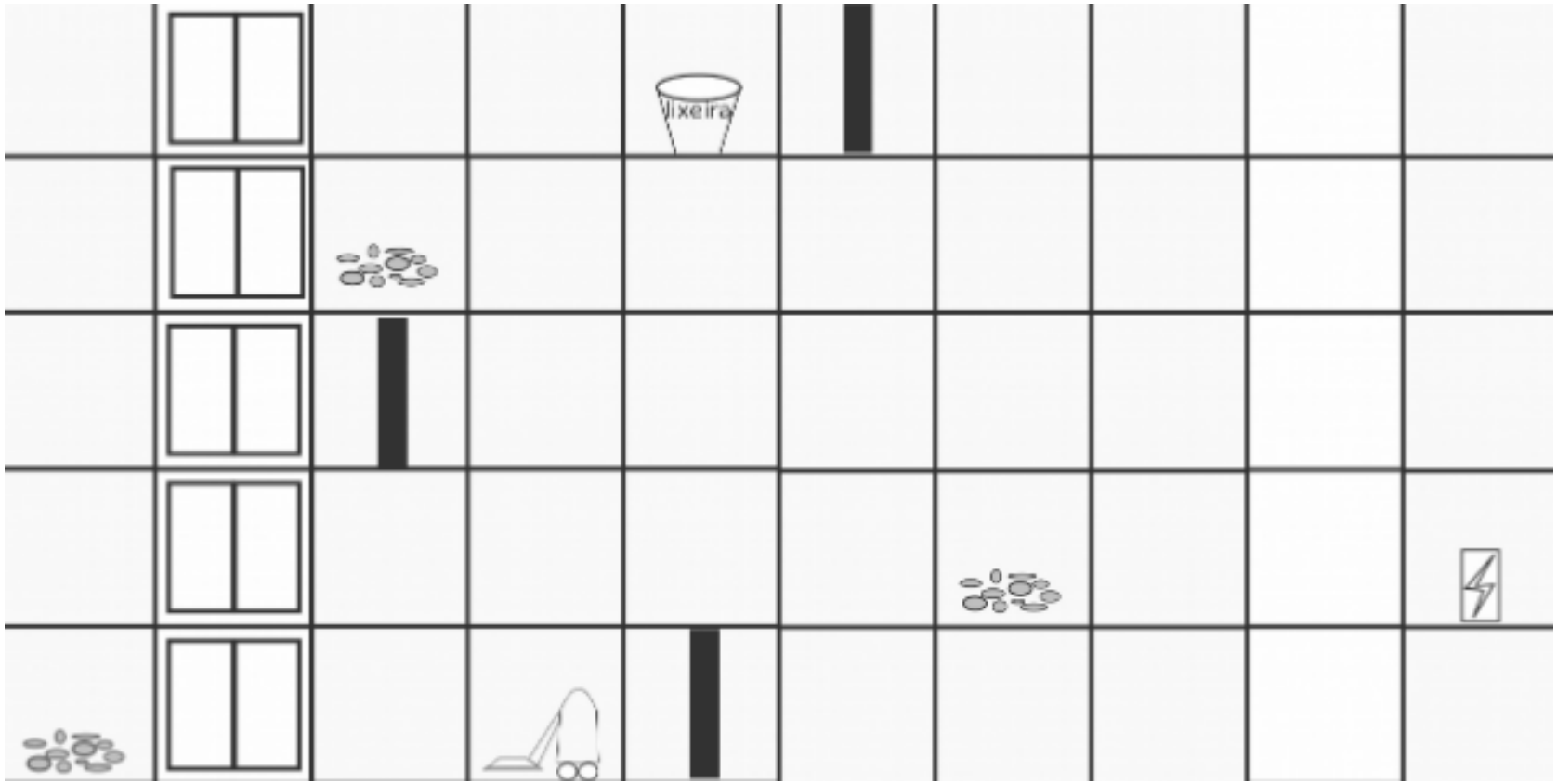
3. Execução - Cenário 2



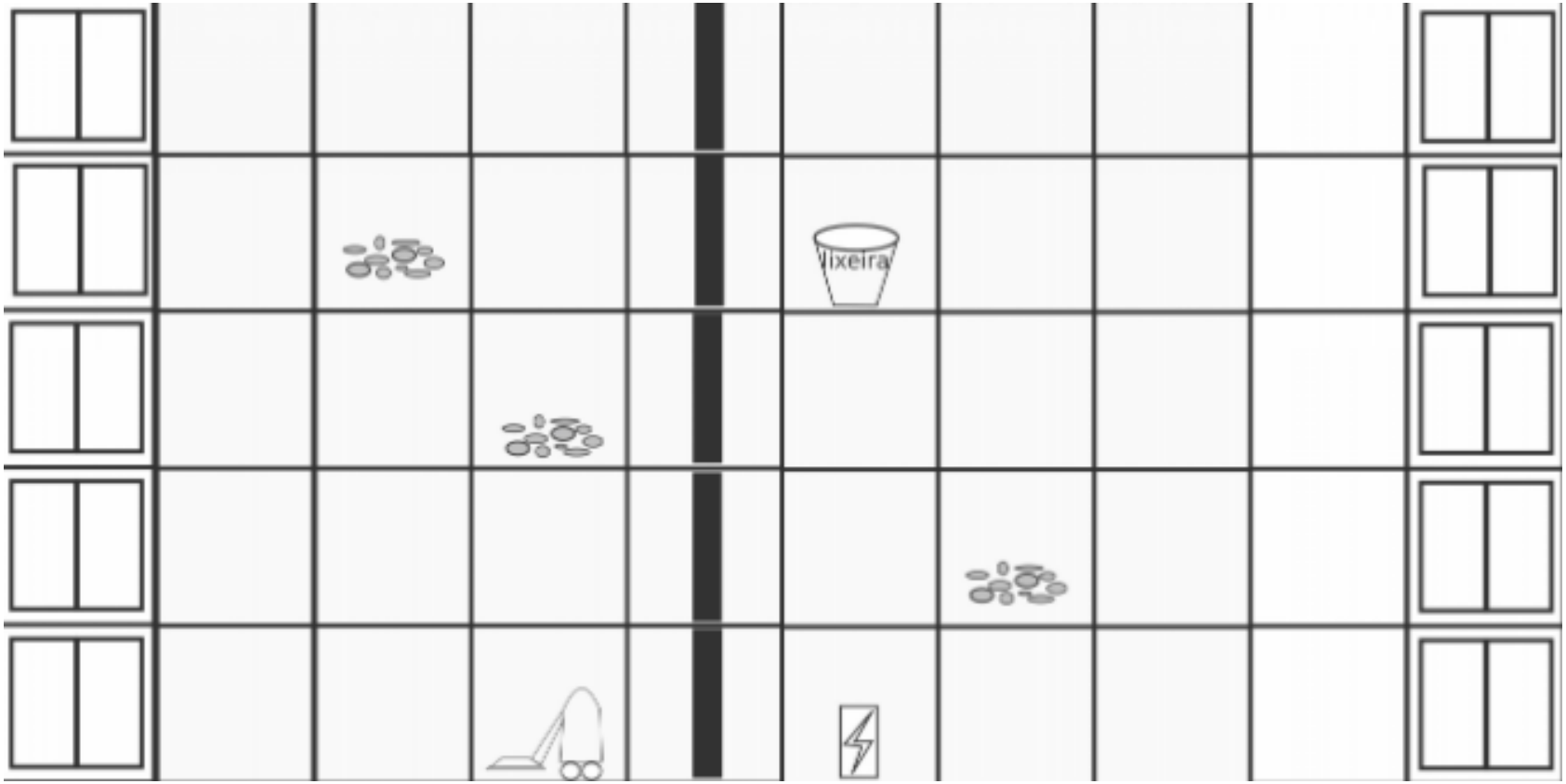
Fonte: Baseado no Material de Apoio, professor Dr. Murilo Naldi Coelho.

3. Execução - Cenário 2

```
?- solucao([3,0],Solucao).
Solucao = [[3, 0], [2, 0], [1, 0], [0, 0], [0, 0], [1, 0], [1, 1], [2, 1], [3, 1], [4, 1], [5, 1], [6, 1]
, [6, 1], [5, 1], [4, 1], [3, 1], [2, 1], [1, 1], [1, 2], [1, 3], [1, 4], [2, 4], [3, 4], [4, 4], [4, 4],
 [3, 4], [2, 4], [1, 4], [1, 3], [2, 3], [2, 3], [1, 3], [1, 2], [1, 1], [2, 1], [3, 1], [4, 1], [5, 1],
[6, 1], [7, 1], [8, 1], [9, 1]] .
```



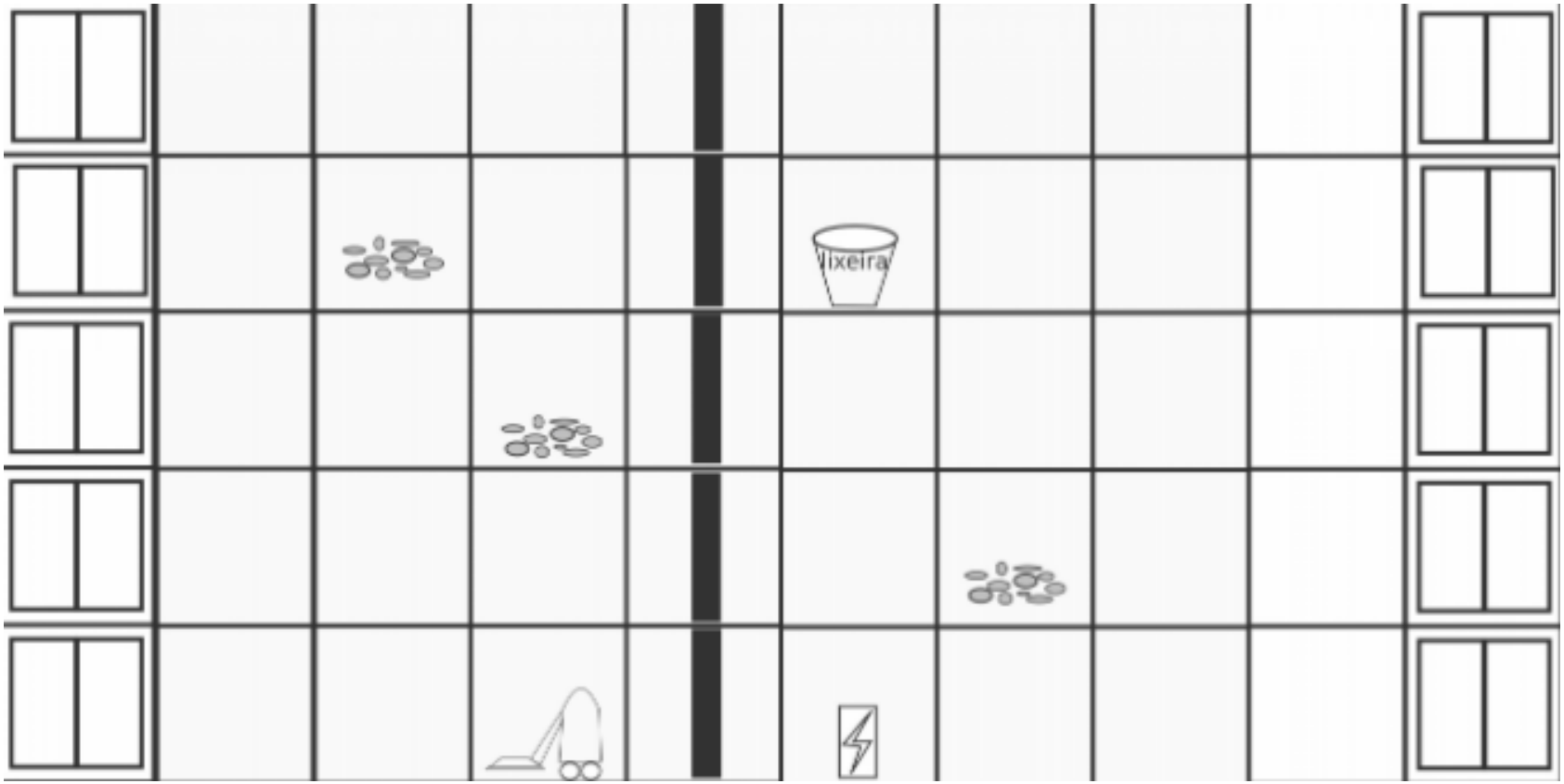
3. Execução - Cenário 3



Fonte: Baseado no Material de Apoio, professor Dr. Murilo Naldi Coelho.

3. Execução - Cenário 3

```
?- solucao([3,0],Solucao).
false.
```



Dúvidas?