

4º Trabalho Prático

Planeamento

Licenciatura em Eng. Informática
Inteligência Artificial



André Baião 48092
Gonçalo Barradas 48402
Guilherme Grilo 48921
Docente: Irene Pimenta

1

Vocabulário

Condições:

aoLado(C, C1) - A casa **C** está ao lado da casa **C1**.

Fluentes:

bloco_na_posicao(B, C) - O bloco **B** está na casa **C**.

bloco_na_mao(B) - Indica que o bloco **B** está na mão do robô.

braco_livre - Indica que a mão do robô está livre.

posicao_livre(C) - Indica que a casa **C** está livre.

robo_na_posicao(C) - Indica que o robô está na casa **C**.

Ações:

agarrar_bloco(B) - O robô vai agarrar o bloco **B**.

largar_bloco(B) - O robô vai largar o bloco **B**.

mover_esquerda(C1, C2) - O robô vai da casa **C2** para a casa **C1**.

mover_direita(C1, C2) - O robô vai da casa **C1** para a casa **C2**.

2

Ação de agarrar no bloco

1	AÇÃO:	agarrar_bloco(Bloco)
2	PRE-CONDIÇÕES:	bloco_na_posicao(Bloco, Casa)
3		robo_na_posicao(Casa)
4		braco_livre
5	AddList:	bloco_na_mao(Bloco)
6		posicao_livre(Casa)
7	DelList:	bloco_na_posicao(Bloco, Casa)
8		braco_livre

Ação de largar o bloco

1	AÇÃO:	largar_bloco(Bloco)
2	PRE-CONDIÇÕES:	bloco_na_mao(Bloco)
3		robo_na_posicao(Casa)
4		posicao_livre(Casa)
5	AddList:	bloco_na_posicao(Bloco, Casa)
6		braco_livre
7	DelList:	bloco_na_mao(Bloco)
8		posicao_livre(Casa)

Ação de mover para a esquerda

1	AÇÃO:	mover_esquerda(CasaNova, CasaAtual)
2	PRE-CONDIÇÕES:	robo_na_posicao(CasaAtual)
3		aoLado(CasaNova, Casa Atual)
4	AddList:	robo_na_posicao(CasaNova)
5	DelList:	robo_na_posicao(CasaAtual)

Ação de mover para a direita

```
1  AÇÃO: mover_esquerda(CasaAtual, CasaNova)
2  PRE-CONDIÇÕES: robo_na_posicao(CasaAtual)
3                  aoLado(CasaAtual, CasaNova)
4  AddList: robo_na_posicao(CasaNova)
5  DelList: robo_na_posicao(CasaAtual)
```

3**Estado Inicial:**

```
1  estado_inicial([
2      em(A, 0),
3      em(B, 1),
4      em(C, 2),
5      em(robo, 0),
6      posLivre(3),
7      maoLivre(),
8      aoLado(0, dir, 1),
9      aoLado(1, dir, 2),
10     aoLado(2, dir, 3),
11     aoLado(3, esq, 2),
12     aoLado(2, esq, 1),
13     aoLado(1, esq, 0)
14 ]).
```

Estado Final:

```
1  estado_final([
2      posLivre(0),
3      em(c, 1),
4      em(a, 2),
5      em(b, 3),
6      em(robo, 0),
7      maoLivre()
8  ]).
```

4**Solução**

```
1  mover_direita(0, 1)
2  agarrar_bloco(b)
3  mover_direita(1, 2)
4  mover_direita(2, 3)
5  largar_bloco(b)
6  mover_esquerda(2, 3)
7  agarrar_bloco(c)
8  mover_esquerda(1, 2)
9  largar_bloco(c)
10 mover_esquerda(0, 1)
11 agarrar_bloco(a)
12 mover_direita(0, 1)
13 mover_direita(1, 2)
14 largar_bloco(a)
15 mover_esquerda(1, 2)
16 mover_esquerda(0, 1)
```

5

Com o objetivo de reduzir o tempo de execução do pop, foi alterado o estado final. O estado final passou a ser o seguinte:

```
1 estado_final([
2 robo_na_posicao(3),
3 bloco_na_posicao(c, 3),
4 bloco_na_posicao(a, 0),
5 bloco_na_posicao(b, 1),
6 posicao_livre(2),
7 braco_livre
8 ]).
```

Recorrendo ao POP para resolver o problema, obtivemos os seguintes conjuntos:

Passos até a solução

```
1 s1-inicial,
2 s5-mover_direita(0,1),
3 s4-mover_direita(1,2),
4 s7-agarrar_bloco(c),
5 s3-mover_direita(2,3),
6 s6-largar_bloco(c),
7 s2-final
```

Links e ordem dos passos

	links	ordem
2	link(s7, posicao_livre(2), s2)	s6<s7<s2
3	link(s4, robo_na_posicao(2)	s4<s3<s7
