UNIVERSIDADE
DE ÉVORA

Relatório do 4º trabalho
Inteligência Artificial

Trabalho realizado por:

- Henrique Rosa, nº 51923

- Diogo Matos, nº 54466

13/06/2024

1 & 2 - Vocabulário e notação STRIPS

Condições:

- **sobre(B1 , B2)** → O bloco B1 está sobre o bloco B2.
- **tamanho(B, X)** → O bloco B tem tamanho X. Serve para distinguir entre blocos grandes e pequenos. Define-se X=1 para os blocos pequenos, e X=2 para os blocos grandes
- **topo(B)** → O bloco B está no topo de uma pilha (não está debaixo de nenhum outro bloco).
- **livre(M)** → A mão M está livre.
- **namao(B,M)** → O bloco B está na mão M.

Ações:

Agarra(B,M) → O robot agarra o bloco B com a mão M, caso esta esteja livre, o bloco seja pequeno e o bloco esteja no topo de uma pilha.

Pré Condições	Add List	Delete List
- tamanho(B,1) - topo(B) - livre(M) - sobre(B,BS)	- namao(B,M) - topo(BS)	- sobre(B,BS) - topo(B) - livre(M)

AgarraG(B) → O robot agarra o bloco B com as duas mãos, caso estas estejam livres, o bloco seja grande e esteja no topo de uma pilha

Pré Condições	Add List	Delete List
- tamanho(B,2) - topo(B) - livre(1) - livre(2) - sobre(B,BS)	- namao(B,1) - namao(B,2) - topo(BS)	- sobre(B,BS) - topo(B) - livre(1) - livre(2)

Larga(B,M,BS) → O robot larga o bloco pequeno B da mão M em cima do bloco BS caso o bloco B esteja na mão M e o Bloco Bs esteja no topo e seja maior ou igual em tamanho ao bloco B

Pré Condições	Add List	Delete List
- topo(BS) - namao(B,M) - tamanho(B,1)	- topo(B) - sobre(B,BS) - livre(M)	- topo(BS) - namao(B,M)

LargaG(B,BS) → O robot larga o bloco grande B em cima do bloco BS caso o bloco B esteja nas duas mãos e o Bloco Bs esteja no topo e seja maior ou igual em tamanho ao bloco B

Pré Condições	Add List	Delete List
- topo(BS) - namao(B,1) - namao(B,2) - tamanho(B,2) - tamanho(BS,2)	- topo(B) - sobre(B,BS) - livre(1) - livre(2)	- topo(BS) - namao(B,1) - namao(B,2)

Ações representadas em prolog:

```
accao(agarra(B,M),  [tamanho(B,1),topo(B),livre(M),sobre(B,BS)],
                    [namao(B,M),topo(BS)],
                    [sobre(B,BS),topo(B),livre(M)]):-   member(B,[a,b,c,d,e]),
                                                        member(BS,[a,b,c,d,e]),
                                                        B\=BS,
                                                        member(M,[1,2]).

accao(agarraG(B),   [tamanho(B,2),topo(B),livre(1),livre(2),sobre(B,BS)],
                    [namao(B,1),namao(B,2),topo(BS)],
                    [sobre(B,BS),topo(B),livre(1), livre(2)]):- member(B,[a,b,c,d,e]),
                                                                member(BS,[a,b,c,d,e]),
                                                                B\=BS.

accao(larga(B,M,BS),[topo(BS),namao(B,M),tamanho(B,1)],
                    [topo(B),sobre(B,BS),livre(M)],
                    [topo(BS),namao(B,M)]):-member(B,[a,b,c,d,e]),
                                            member(BS,[a,b,c,d,e]),
                                            B\=BS,
                                            member(M,[1,2]).

accao(largaG(B,BS) ,[topo(BS),namao(B,1),namao(B,2),tamanho(B,2),tamanho(BS,2)],
                    [topo(B),sobre(B,BS),livre(1),livre(2)],
                    [topo(BS),namao(B,1),namao(B,2)]):- member(B,[a,b,c,d,e]),
                                                       member(BS,[a,b,c,d,e]),
                                                       B\=BS.
```

3 - Estado inicial e final em prolog

```
estado_inicial([tamanho(a,2), tamanho(b,1), tamanho(c,1), tamanho(d,2), tamanho(e, 1),
                topo(e), topo(c),
                tamanho(chao,2), topo(chao),
                sobre(a,chao), sobre(b,chao),
                sobre(d,a), sobre(e,d), sobre(c,b),
                livre(1), livre(2) ]).

estado_final([  tamanho(a,2), tamanho(b,1), tamanho(c,1), tamanho(d,2), tamanho(e, 1),
                topo(e), topo(a),
                tamanho(chao,2), topo(chao),
                sobre(c,chao), sobre(d,chao),
                sobre(e,b), sobre(b,c),sobre(a,d),
                livre(1), livre(2) ]).
```

Depois da definição ações, estas foram testadas uma a uma, com estados iniciais e finais diferentes, formando problemas mais pequenos e simples, que necessitariam apenas de uma condição para serem resolvidos. Isto foi feito com o propósito de assegurar que todas as ações estão corretamente definidas:

Ação	Estados	Output
agarra(B,M)	`estado_inicial([tamanho(a,2), tamanho(d,2), tamanho(chao,2), topo(chao), sobre(a,chao), sobre(d,a), topo(d), livre(1), livre(2)]).` `estado_final([ topo(chao), topo(a), sobre(a,chao), namao(d,1), namao(d,2) ]).`	`P = [s1-inicial,s3-agarra(d,1),s2-final] ?`
agarraG(B)	`estado_inicial([tamanho(a,2), tamanho(d,2), tamanho(chao,2), topo(chao), sobre(a,chao), sobre(d,a), topo(d), livre(1), livre(2)]).` `estado_final([ topo(chao), topo(a), sobre(a,chao), namao(d,1), namao(d,2) ]).`	`P = [s1-inicial,s3-agarraG(d),s2-final] ?`
larga(B,M,BS)	`estado_inicial([tamanho(a,2), tamanho(d,1), tamanho(chao,2), topo(chao), sobre(a,chao), topo(a), namao(d,1), livre(2)]).` `estado_final([ topo(chao), topo(d), sobre(a,chao), sobre(d,a), livre(1), livre(2) ]).`	`P = [s1-inicial,s3-larga(d,1,a),s2-final] ?`
largaG(B.BS)	`estado_inicial([tamanho(a,2), tamanho(d,2), tamanho(chao,2), topo(chao), sobre(a,chao), topo(a), namao(d,1), namao(d,2) ]).` `estado_final([ topo(chao), topo(d), sobre(a,chao), sobre(d,a), livre(1), livre(2) ]).`	`P = [s1-inicial,s3-largaG(d,a),s2-final] ?`

Obs: Um dos erros que cometemos nesta fase foi o de não otimizar o valor de M, sendo esse valor o número de passos do plano. Inicialmente foi deixado a 12 como um valor maior do que qualquer valor necessário, mas isto fazia com que o pop encalhasse na primeira ação do código e acabasse por dar stack overflow na sua execução. Acabamos por diminuir o valor de 12 para 3, para fazer os testes de cada uma das ações, e funcionou perfeitamente, como vemos na tabela acima

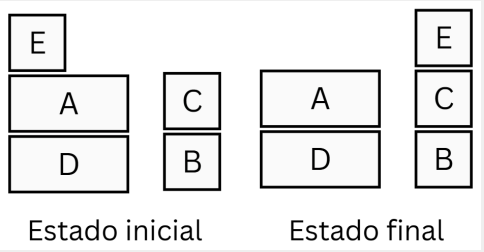
4 - Resolução do POP:

Depois dos testes das ações, e de percebermos que todas elas estavam a fazer o que era suposto, tentámos resolver o problema com o ficheiro pop.pl dado nas aulas práticas, mas a execução retornou "Fatal Error: Atom table full", ou seja, não conseguimos obter o plano do pop desta forma.

Como tal, iremos elaborar problemas mais simples, com menos passos, de modo a obter o seu planeamento. Como já vimos pelos testes das várias ações definidas, o programa suporta planos com apenas um passo, como tal vamos começar pelos dois passos.

Todas as execuções vão ser feitas mudando o "M" do predicado **escolheSk/6** para menor que n+2, sendo n o número de passos esperado para a resolução.

Dois passos (M < 4):

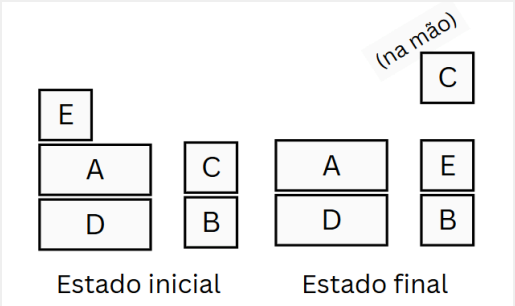



```
estado_inicial([tamanho(a,2), tamanho(b,1), tamanho(c,1), tamanho(d,2), tamanho(e, 1),
                topo(e), topo(c),
                tamanho(chao,2), topo(chao),
                sobre(e,a), sobre(a,d), sobre(d,chao),
                sobre(c,b),sobre(b,chao),
                livre(1), livre(2) ]).

estado_final([  tamanho(a,2), tamanho(b,1), tamanho(c,1), tamanho(d,2), tamanho(e, 1),
                tamanho(chao,2), topo(chao),
                topo(a), topo(e),
                sobre(c,b), sobre(e,c), sobre(d,chao),
                sobre(b,chao),sobre(a,d),
                livre(1), livre(2) ]).
```

Output: `P = [s1-inicial,s3-agarra(e,1),s272-larga(e,1,c),s2-final] ?`

Três passos (M<5):

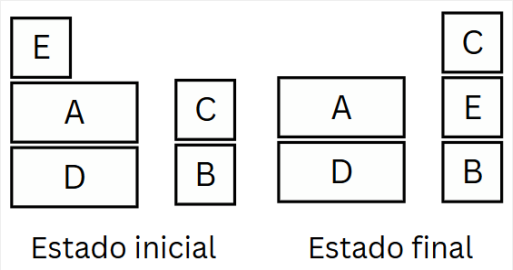



```
estado_inicial([tamanho(a,2), tamanho(b,1), tamanho(c,1), tamanho(d,2), tamanho(e, 1),
                topo(e), topo(c),
                tamanho(chao,2), topo(chao),
                sobre(e,a), sobre(a,d), sobre(d,chao),
                sobre(c,b),sobre(b,chao),
                livre(1), livre(2) ]).

estado_final([  tamanho(a,2), tamanho(b,1), tamanho(c,1), tamanho(d,2), tamanho(e, 1),
                tamanho(chao,2), topo(chao),
                topo(a), topo(e),
                sobre(e,b), sobre(b,chao),
                sobre(a,d),sobre(d,chao),
                namao(c,1), livre(2) ]).
```

Output: `P = [s1-inicial,s3-agarra(e,2),s9344-agarra(c,1),s9076-larga(e,2,b),s2-final] ?`

Quatro passos (M<6):



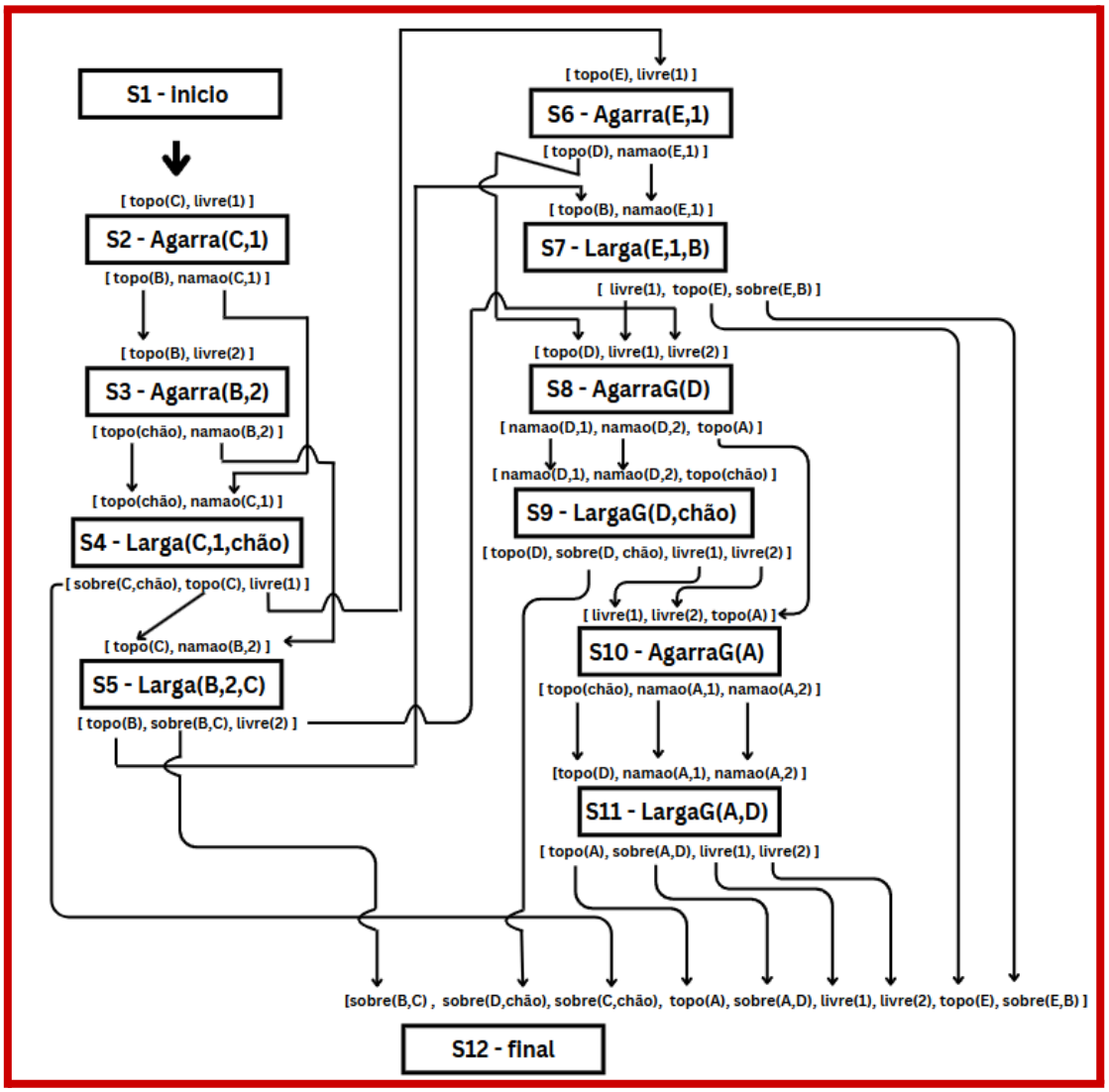

```
estado_inicial([tamanho(a,2), tamanho(b,1), tamanho(c,1), tamanho(d,2), tamanho(e, 1),
                topo(e), topo(c),
                tamanho(chao,2), topo(chao),
                sobre(e,a), sobre(a,d), sobre(d,chao),
                sobre(c,b),sobre(b,chao),
                livre(1), livre(2) ]).

estado_final([  tamanho(a,2), tamanho(b,1), tamanho(c,1), tamanho(d,2), tamanho(e, 1),
                tamanho(chao,2), topo(chao),
                topo(a), topo(c),
                sobre(c,e), sobre(e,b), sobre(b,chao),
                sobre(a,d),sobre(d,chao),
                livre(1), livre(2) ]).
```

Output:

```
Fatal Error: Atom table full (max atom: 32768, environment variable used: MAX_ATOM)
```

Como podemos ver, para a nossa implementação, o pop não consegue planear a resolução de um problema com mais do que três passos. Como tal, vamos proceder a fazer manualmente o plano do pop para o problema apresentado no enunciado, com as ameaças e a sua resolução. Ignoramos as condições sobre o tamanho dos blocos, pois essas condições não mudam em toda a execução do código, e também não fizemos alusão às condições satisfeitas pelo estado inicial. tomamos estas decisões para manter a simplicidade e facilidade de visualização do esquema que fizemos:



Ameaça	Ameaçado	Condição	Resolução
S11	S8	Topo(D)	despromoção de S8, S8<S11
S10	S2	livre(1)	promoção de S10, S2<S10
S10	S6	livre(1)	promoção de S10, S6<S10
S10	S8	livre(1)	promoção de S10, S8<S10
S10	S3	livre(2)	promoção de S10, S3<S10
S10	S8	livre(2)	promoção de S10, S8<S10
S9	S4	topo(chão)	promoção de S9, S4<S9
S8	S11	topo(D)	despromoção de S8, S8<S11
S8	S2	livre(1)	promoção de S8, S2<S8
S8	S6	livre(1)	promoção de S8, S6<S8
S8	S10	livre(1)	despromoção de S8, S8<S10
S8	S3	livre(2)	promoção de S8, S3<S8
S8	S10	livre(2)	despromoção de S8, S8<S10
S7	S3	topo(B)	promoção de S7, S3<S7
S6	S2	livre(1)	promoção de S6, S2<S6
S6	S8	livre(1)	despromoção de S6, S6<S8
S6	S10	livre(1)	despromoção de S6, S6<S10
S5	S2	topo(C)	promoção de S5, S2<S5
S4	S9	topo(chão)	despromoção de S4, S4<S9
S3	S6	topo(B)	despromoção de S3, S3<S6
S3	S8	livre(2)	despromoção de S3, S3<S8
S3	S10	livre(2)	despromoção de S3, S3<S10
S2	S5	topo(C)	despromoção de S2, S2<S5
S2	S6	livre(1)	despromoção de S2, S2<S6
S2	S8	livre(1)	despromoção de S2, S2<S8
S2	S10	livre(1)	despromoção de S2, S2<S10