Planejador para Empilhar Blocos de Diferente Dimensões

Alunos: Natanael Bezerra de Oliveira (22152258), Lucas Silva Araújo (221538899), Washington Antônio Moreno Riega (22152254)

Q1) Proponha uma linguagem de representação para esta instância do mundo dos blocos que inclua detalhes como: tamanho do bloco; posição na mesa com sua medida de espaço lateral e vertical; condições de: vacância para um espaço ser preenchido por bloco (vertical e horizontal), estabilidade para um bloco estar sobre outro (vacância no topo e centro de massa em caso de bloco maior sobre menor). Justifique sua proposta comparando cada elemento (bloco, lugar, objeto, etc) com Figure 17.1 do cap 17.

Nossa proposta é fundamentada no plano cartesiano, empregando coordenadas x e y para dispor os elementos na mesa.

```
% Blocks definition
block(a).
block(b).
block(c).
block(d).

% Sizes
size(a, 1).
size(b, 1).
size(c, 2).
size(d, 3).

% Places definition
place(1).
place(2).
place(3).
place(4).
place(5).
place(6).
```

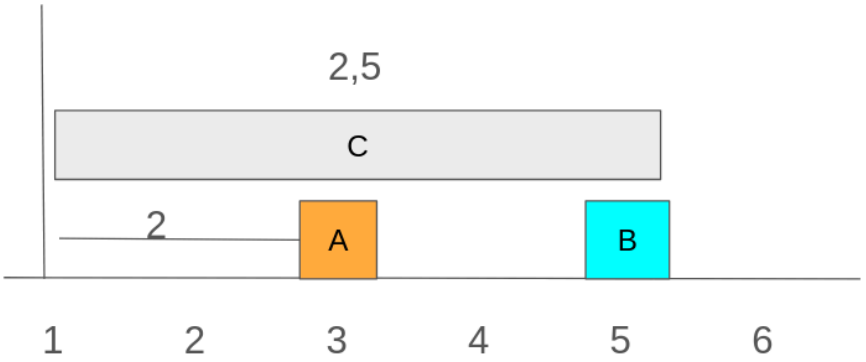
Utilizamos os axiomas block, que define o que constitui um bloco; size, que determina as dimensões de um bloco (na nossa representação, consideramos apenas a largura dos blocos); e place, que define um espaço. Na verdade, em nossa representação, os places estabelecem um plano cartesiano. No exemplo fornecido, o place(1) é combinado com outros places para gerar as coordenadas y.

```
state1([ on(c, at(1, 1)), dirty(2,1), on(a, at(4, 1)), on(b, at(6, 1)), on(d, at(4, 2)), dirty(5,2), dirty(6,2) ]).
```

Para a representação do mundo dos blocos, fazemos uso do predicado on(Block, at(X,Y)), que marca o início de um bloco. Quando o bloco possui tamanho superior a 1, todos os espaços que ele ocupa são assinalados como dirty(X,Y). Optei por representar o que está

"sujo" como uma maneira mais simples de descrever o estado, em contraste com a representação do que está limpo.

Para determinar a estabilidade de um bloco, incorporamos o conceito de "bloco âncora". Por exemplo, ao posicionar um bloco C na coordenada at(1,2) com um tamanho de 5, mapeamos a posição abaixo dele em busca de dois blocos âncora, que serão responsáveis por assegurar a estabilidade. A estabilidade é alcançada quando a distância entre o extremo do bloco C e o extremo do bloco âncora é inferior à metade do tamanho do bloco. Dessa forma, garantimos que a configuração esteja estável.



Em prolog ficou:

```
first_anchor([Ac | _], Coords) :-
    (Ac = dirty(X, Y) ; Ac = on(_, at(X, Y))),
    Coords = (X-Y), !.

first_anchor([_ | Tail], Coords) :-
    first_anchor(Tail, Coords).

is_possible(Block, (Xr-_), (Xa-_)) :-
    size(Block, Size),
    Med is Size/2,
    E is abs(Xa-Xr),
    E < Med.
```

Q2) Modificar o código do planner da figura 17.6 de tal maneira que este manipule corretamente variáveis sobre goals e também ações conforme discussão na sessão 17.5. Indique esta mudança com a explicação.

```
can(Block, [], []).

can(Block,Situation,Positions) :-
    nth0(0, Positions, First), last(Positions, Last),
    first_anchor(Situation, Coords_1), % Busy or Block more left
    reversed(Situation, SituationRev),
    first_anchor(SituationRev, Coords_2), %Busy or Block more rigth
    is_possible(Block,First, Coords_1),
    is_possible(Block,Last, Coords_2).
```

A função can/3 é responsável por determinar se é possível executar uma ação com um bloco específico, levando em consideração a situação do ambiente, representada pelo predicado is_possible/3.

```
have_space(Block, X) :-
    size(Block, Size),
    F is X + Size - 1,
    place(F).
```

Adicionalmente, foram introduzidos predicados para verificar a viabilidade de posicionar um bloco em uma determinada posição, através do novo predicado.

```
goal(State, move(Block, at(X,Y))) :-
    place(X), place(Y),
    block(Block),
    have_space(Block, X),
    generate_positions(Block, X, Y, Range), % Position at the block will occuped
    look_positions(State, Range, Situation),
    busy_all(State, Situation),
    member( on(Block, at(X_1, Y_1)), State ),
    Y_b is Y_1+1,
    \+ busy( X_1, Y_b, State), % Don't can have nothing above.
    stabilize(State, Block, X,Y).
    can(Block,Situation, Range).
      NB02001, 9 hours ago • Goal completly.
```

E o predicado goal verifica se uma ação é factível.

Q3)

<u>Situação 1:</u>

movimentos em alto nível:

primeiro move bloco 'a' de 3-4 para 0-1 (sobe 1)
segundo move bloco 'd' de 3-6 para 2-5 (desce 1)

state(Initial), finalState(Final), plan(Initial, Final, Plan).

Initial = [on(a, at(3,0)), dirty(4,0), on(b, at(5,0)), dirty(6,0), on(c, at(0,0)), dirty(1,0), dirty(2,0), on(d, at(3,1)), dirty(4,1), dirty(5,1), dirty(6,1)],

Final = [on(a, at(0,1)), dirty(1,1), on(b, at(5,0)), dirty(6,0), on(c, at(0,0)), dirty(1,0), dirty(2,0), on(d, at(2,0)), dirty(3,0), dirty(4,0), dirty(5,0)],

Plan = [move(b, at(3,1)), move(a, at(2,1)), move(c, at(4,1), move(b, at(5,2), move(a, at(4,2), move(d, at(3,0))] .

Situação 1(a):
movimentos:

primeiro move bloco 'b' de 5-6 para 1-2 (sobe 1)
segundo move bloco 'd' de 2-5 para 3-6
terceiro move bloco 'b' de 1-2 para 5-6
quarto move bloco 'a' de 0-1 para 4-5
quinto move bloco 'c' de 0-2 para 4-6 (sobe 2)

state(Initial), finalState(Final), plan(Initial, Final, Plan).

Initial = [on(a, at(0,1)), dirty(1,1), on(b, at(5,0)), dirty(6,0), on(c, at(0,0)), dirty(1,0), dirty(2,0), on(d, at(2,0)), dirty(3,0), dirty(4,0), dirty(5,0)],

Final = [on(a, at(4,1)), dirty(5,1), on(b, at(5,1)), dirty(6,1), on(c, at(4,2)), dirty(5,2), dirty(6,2), on(d, at(3,0)), dirty(4,0), dirty(5,0), dirty(6,0)],

Plan = [move(b, at(1,1)), move(d, at(3,0)), move(b, at(5,1), move(a, at(4,1), move(c, at(4,2)].

Situação 1(b):

movimentos:

primeiro move bloco 'b' de 5-6 para 3-4 (sobe 1)
segundo move bloco 'a' de 0-1 para 2-3
terceiro move bloco 'c' de 0-2 para 4-6 (sobe 1)
quarto move bloco 'b' de 3-4 para 5-6 (sobe 1)
quinto move bloco 'a' de 2-3 para 4-5 (sobe 1)
sexto move bloco ‘d’ de 2-5 para 3-6

state(Initial), finalState(Final), plan(Initial, Final, Plan).

Initial = [on(a, at(0,1)), dirty(1,1), on(b, at(5,0)), dirty(6,0), on(c, at(0,0)), dirty(1,0), dirty(2,0), on(d, at(2,0)), dirty(3,0), dirty(4,0), dirty(5,0)],

Final = [on(a, at(4,2)), dirty(5,2), on(b, at(5,2)), dirty(6,2), on(c, at(4,1)), dirty(5,1), dirty(6,1), on(d, at(3,0)), dirty(4,0), dirty(5,0), dirty(6,0)],

Plan = [move(b, at(3,1)), move(a, at(2,1)), move(c, at(4,1), move(b, at(5,2), move(a, at(4,2), move(d, at(3,0))] .

Situação 1(c):

movimentos:

primeiro move bloco 'a' de 0-1 para 5-6
segundo move bloco 'd' de 2-5 para 0-3 (sobe 1)
terceiro move bloco 'a' de 5-6 para 2-3 (desce 1)

state(Initial), finalState(Final), plan(Initial, Final, Plan).

Initial = [on(a, at(0,1)), dirty(1,1), on(b, at(5,0)), dirty(6,0), on(c, at(0,0)), dirty(1,0), dirty(2,0), on(d, at(2,0)), dirty(3,0), dirty(4,0), dirty(5,0)],

Final = [on(a, at(2,0)), dirty(3,0), on(b, at(5,0)), dirty(6,0), on(c, at(0,0)), dirty(1,0), dirty(2,0), on(d, at(0,1)), dirty(1,1), dirty(2,1), dirty(3,1)],

Plan = [move(a, at(5,1)), move(d, at(0,1)), move(a, at(2,0))] .

Situações 2 e 3

Considere as situações 2 e 3, e faça uma análise do que é necessário para seu planejador gerar planos de ações de S0 até o estado final de cada situação. Isso deve levar em conta tanto sua proposta de linguagem quanto padrões de situações dos blocos.

Por ter muitos espaços o espaço de busca é muito grande, sendo necessário pensar em uma solução mais eficiente, entretanto como todas as nossas definições de estabilidade e espaço acreditamos que em algum momento nosso planejador encontraria a solução.

```
% Sitaução 2
Plan = [move(b, at(3,1)), move(a, at(3,2)), move(c, at(5,2)), move(a, at(5,3)), move(b, at(6,3)) ]
```

```
% Sitaução 3

Plan = [move(d, at(1,2)), move(a, at(6,2)), move(4, at(3,1)), move(a, at(1,2)), move(b, at(2,2)), move(d, at(4,1)) ]
```