**Prática: resolução de problemas mediante busca em Prolog**

Requisitos:

1. Trabalho com listas em Prolog
   1. M. Teresa Escrig, pg. 21 - 31
   2. Bratko, pg. 64 - 78
2. Notação operador
   1. M. Teresa Escrig, pg. 42 - 43
   2. Bratko, pg. 78 - 84

**Introdução:**

Nesta prática são apresentados vários problemas de busca num espaço de estados. Para a solução deste tipo de problemas existem várias aproximações em termos de métodos e trataremos de utilizar o método de *busca em profundidade primeiro*.

Este método se programa facilmente no estilo de programação recursiva do Prolog. Mais ainda, o Prolog ao tratar de satisfazer os objectivos emprega a busca em profundidade primeiro.

A implementação deste algoritmo tem como base a ideia de que para encontrar uma sequência solução de um problema, **Sol**, a partir de um nó dado, **N**, até outro nó objectivo:

* Se **N** é um nó objectivo, então **Sol** = [**N**], ou
* Se existe um nó sucessor de **N**, **N1**, tal que existe um caminho **Sol1** desde **N1** até um nó objectivo, então **Sol** = [**N**|**Sol1**].

Traduzido a Prolog:

*resolver(N, [N]) :- objectivo(N).*

*resolver(N, [N|Sol1]) :-*

*s(N, N1), resolver(N1, Sol1).*

Ver Bratko, pg. 251 – 256.

**Exercício 1:**

O problema da geração de planos num mundo dos blocos consiste em encontrar uma sequência de acções (um plano) que permite reorganizar uma pilha de blocos, para se passar de uma situação inicial à outra final, na qual os blocos são dispostos da forma desejada (no nosso caso, como se mostra na figura). Devem cumprir-se as seguintes restrições: só se pode mover um bloco de cada vez; um bloco pode ser movido se não houver outro bloco sobre ele; um bloco pode ser colocado sobre outro bloco (mas não sobre si mesmo) ou em certas posições da mesa. Para simplificar,só se admite a possibilidade de colocar os blocos em três posições diferentes, designadas abaixo P1, P2, e P3.



[[c,a,b], [], []]

---------- --- ---

P1 P2P3

Cada estado é representado por uma lista de listas, onde cada lista corresponde à pilha de blocos relativa a uma das posições, P1, P2 ou P3, por exemplo [[c a b], [], []].

De acordo a esta representação qualquer dos estados seguintes constituiria um objectivo, [[a b c], [], []], [[], [a b c], []], [[], [], [a b c]].

O problema é resolvido através da pergunta "?- *resolver(Inicial, Solução).*", onde *Inicial* define o estado inicial.

1. Carregue o ficheiro *blocos.pl* e analise a solução.
2. Determine a solução correspondente a um estado inicial determinado.

A solução proposta é pouco eficiente e por isso pode ter ocorrido um erro de desbordamento da pilha, devido a geração de ciclos no processo de solução. No ficheiro *blocos1.pl* se implementa uma solução onde se controla a geração de estados repetidos. A referida solução pode ser vista nas páginas 253 – 254 do manual de Bratko.

1. Carregue o referido ficheiro.
2. Tente novamente encontrar algumas soluções.

**Exercício 2:**

Há um macaco situado na entrada de uma sala. No centro da sala existe uma banana pendurada no teto. A posição da banana é tão alta que esta não pode ser alcançada a partir do solo (por muito que salte o macaco). Ao lado da janela, do lado oposto da sala, há uma caixa que o macaco pode utilizar. O macaco pode realizar as seguintes acções: *caminhar ao redor da sala*, *empurrar a caixa* (e deslocá-la de um lugar para outro), *subir na caixa* e *pegar a bananas* se caixa estiver directamente por baixo da mesma. Pode o macaco pegara banana?

Neste problema o estado pode representar-se através de um termo s(Pm, Sm, Pc, Tm), no qual cada argumento significa:

* Pm: o macaco está na posição Pm (porta, janela ou centro);
* Sm: o macaco se encontra sobre (solo ou caixa);
* Pc: posição em que se encontra a caixa (porta, janela ou centro);
* Tm: o macaco tem ou não a banana (tem, naotem).

As acções ou movimentos possíveis se especificam através dos seguintes termos e constantes:

* andar(P1, P2): o macaco anda da posição P1 à posição P2;
* empurrar(P1, P2): o macaco empurra a caixa da posição P1 à P2;
* subir: o macaco sobe para cima da caixa;
* agarrar: o macaco agarra a banana.

São definidos dois predicados que se descrevem a continuação:

* possivel(S): indica que o estado S é possível;
* move(S1, M, S2): permite passar de um estado a outro, quando se realiza uma acção ou movimento; o significado do predicado é que o movimento M faz com que o estado S1 mude para S2.

1. Carregue o ficheiro *macaco.pl* no ambiente e faça um breve estudo da organização da solução.
2. Compile o ficheiro e faça diversas perguntas ao sistema.
3. Verifique se é possível o estado final. O estado final pode ser representado por s(\_, \_, \_, tem).
4. Experimente verificar se são possíveis outros estados.
5. Pergunte quais são as acções possíveis (“?- move(\_, A, \_).”).

A solução apresentada permite saber se determinado estado é possível mas não gera uma trajectória até à mesma. No ficheiro *macaco1.pl* se implementa uma solução que permite mostrar a referida trajectória.

1. Carregue o ficheiro e analise a solução. Na mesma "[]" e "[M|Sol1]" têm a ver com o manejo de estruturas em forma de lista. Leia o conteúdo correspondente e trate de interpretar as linhas de código nas quais aparecem.
2. Consulte a solução obtida pelo sistema para um estado inicial determinado. Por exemplo “?- resolve(s(porta, solo, janela, naotem), Sol).”.
3. Altere o estado inicial e verifique as soluções encontradas.

**Exercício 3:**

Suponhamos que temos um tanque com água suficiente e dois recipientes de 7 e 5 litros, inicialmente vazios. Encontre uma sequência de acções que deixe 4 litros de água no recipiente de 7 litros. As acções legais são: encher um recipiente, tomando a água do tanque; esvaziar um recipiente no tanque; transferira água de um recipiente para outro (até que o recipiente desde o qual se transfere esteja vazio); transferira água de um recipiente para outro (até que o recipiente para o qual a mesma é transferida esteja cheio).

Neste caso a noção de estado se representa através de um vector com duas coordenadas est(C7, C5), no qual a variável C7 representa a quantidade de água existente no recipiente de 7 litros e C5 a quantidade no recipiente de 5 litros.

Os termos que caracterizam as acções, encher, esvaziar e transferir, se implementam utilizando-se construtores binários. O segundo argumento do termo representa a quantidade com que se enche, esvazia ou é transferida no caso de um recipiente dado.

1. Carregue o programa correspondente (*contentores1.pl*) e analise a solução.
2. Verifique a solução para um estado inicial determinado, por exemplo “est(0, 0)”.
3. Experimente outros estados alvos e verifique as correspondentes soluções.
4. Altere o método de busca adicionando controlo de repetições.