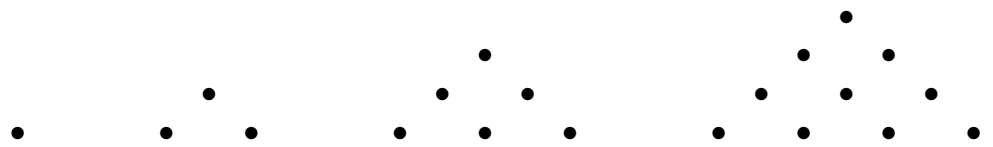
Alguns Exercícios de Inteligência Artificial

Ana Paula Tomás
Nelma Moreira

Departamento de Ciência de Computadores
Faculdade de Ciências, Universidade do Porto
email: {apt,nam}@ncc.up.pt

1997

1. **Números triangulares**



   Um inteiro positivo diz-se *triangular* sse for possível representá-lo como se ilustra acima (exemplos, 1, 3, 6, 10, ...). Defina a sucessão de números triangulares por recorrência.

   (a) Escrever um predicado `triang(N,B)` para verificar se `N` é triangular, sendo `B` a base do triângulo.

   (b) Escrever um predicado `todostriang(I,F)` para determinar todos os números triangulares entre `I` e `F`, dados pelo utilizador.

2. **Problema do Caixeiro Viajante**

   (a) Escrever um predicado `camHam(Verts,In,Fim)` para verificar se existe um caminho hamiltoneano (isto é, um caminho que passa uma e uma só vez por cada vértice), com origem em `In` e fim em `Fim`. Suponha que `Verts` é o conjunto de vértices do grafo, e que os arcos são dados por `arco/2`.

   (b) Modificar o programa anterior para indicar ainda a sequência de vértices no caminho.

   (c) Escrever um predicado para verificar se um dado grafo tem algum ciclo Hamiltoneano, isto é um ciclo que passa uma e uma só vez por cada vértice com excepção do vértice final.

   (d) Suponha que o grafo tem distâncias associadas aos arcos. Escrever um predicado para determinar o ciclo Hamiltoneano cuja soma das distâncias é mínima.

3. **Colorir mapas**. Pretende-se colorir um mapa de forma que as regiões com fronteira comum sejam coloridas com cores diferentes. Suponha que o mapa é representado por `front/2`. Escreva um predicado `colorir(N,L)` em que `N` representa o número mínimo de cores necessárias e `L` é uma lista de termos `t(Regiao,Cor)`.

4. **Quadrados mágicos** Um quadrado mágico $N \times N$ de inteiros positivos é uma matriz quadrada em que a soma dos inteiros em cada linha ou coluna é igual. Escrever um predicado `magic(N,Soma,Quadrado)` que sucede se existir um `Quadrado` mágico $\texttt{N} \times \texttt{N}$ cuja soma dos elementos numa linha/coluna é `Soma`. O quadrado é representado por uma lista de listas, em que cada lista é uma linha.

5. Suponha numa árvore binária cada nó interno é representado por um termo `no(V,E,D)`, sendo as folhas representadas por termos `f(V)`. Suponha ainda que se `E` (respectivamente, `D`) for a lista vazia então a sub-árvore esquerda (respectivamente, direita) não existe. Cada `V` é um valor inteiro. Nos programas seguintes não use o predicado `append`.

   (a) Escrever um predicado `vals(Arv,L)` para construir a lista `L` de valores nos nós e/ou folhas da árvore `Arv` fazendo pesquisa em profundidade da esquerda para a direita.

   (b) Escrever um predicado `floresta(LArvs,Rz,F)` que dada uma floresta (isto é, lista de árvores) constroi a lista de raízes (isto é, nós de topo) e a floresta de subárvores "penduradas" directamente nos nós de topo, da esquerda para a direita.

   (c) Escrever um predicado que dada uma árvore constroi a lista das listas de valores que se encontram no mesmo nível de profundidade.

   (d) Escrever um predicado que dadas duas árvores as reescreve, trocando a folha mais à esquerda na primeira com mais à direita na segunda. Só deve descer cada árvore uma só vez!

6. Suponha dada uma lista de termos `o(C,N)` em que `C` pode ser um dos atómos `azul`, `verde`, ou `branco`. Não use o predicado `append`.

   (a) Escrever um predicado `igseg(L,AVB)` que dada uma lista `L` a reescreve de forma que todos os elementos de cor `azul` apareçam primeiro, seguidos dos de cor `verde`, seguidos dos restantes, mantendo os da mesma cor na ordem de ocorrência em `L`. Percorra a lista apenas uma vez.

   (b) Modificar o predicado que escreveu em 6a) para preceder os elementos da mesma cor do número de elementos nessas condições.

7. Suponha dada uma lista de termos (sem variáveis) em que cada elemento é um átomo ou um functor `p(L)` em que `L` é uma lista do mesmo tipo.

   (a) Escrever uma DCG para traduzir a lista de termos noutra, devendo traduzir também as listas nos termos `p(_)`. Cada sublista `[ee,tt,a]` é traduzida por `[ff,kk]`, e cada sublista `[ee,tt,b]` por `[g,kk,mm]`, mantendo-se as restantes sequências.

   (b) Escrever uma DCG para processar a lista dada de modo a obter a lista de atómos em `L`. Pretende-se que se `L` for

   $$[\texttt{ee}, \texttt{tt}, \texttt{a}, \texttt{p}([\texttt{ee}, \texttt{tt}, \texttt{xx}, \texttt{p}([\texttt{xx}, \texttt{yy}, \texttt{aa}]), \texttt{kk}]), \texttt{tt}, \texttt{a}, \texttt{p}([\texttt{xx}, \texttt{aa}, \texttt{ee}, \texttt{tt}, \texttt{b}])]$$

   a lista resultante seja

   $$[\texttt{ee}, \texttt{tt}, \texttt{a}, \texttt{ee}, \texttt{tt}, \texttt{xx}, \texttt{xx}, \texttt{yy}, \texttt{aa}, \texttt{kk}, \texttt{tt}, \texttt{a}, \texttt{xx}, \texttt{aa}, \texttt{ee}, \texttt{tt}, \texttt{b}].$$

8. Escrever uma DCG para dado um inteiro entre 0 e 1000 escrever por extenso o seu valor. O inteiro é representado pela lista de dígitos.

9. Escrever uma DCG para reescrever um texto incluindo à frente de cada *quantia* o seu valor por extenso. Cada *quantia* é inteira e representada por x\$00, por exemplo, 530\$00, devendo ficar no texto final "530\$00 (quinhentos e trinta escudos)". O texto é representado como uma lista de caracteres.

10. ***Tokens* & Leitura de Ficheiros.** Escreva um programa que a partir dum texto dado como uma lista de caracteres construa uma lista de "tokens" segundo os seguintes critérios:

    - os caracteres `brancos` são ignorados.
    - sequências de caracteres `L` entre dois caracteres `*` ficam num termo Prolog `s(L)`.
    - sequências de letras `L` são agrupadas num termo da forma `id(T)`, onde `T` é o átomo Prolog correspondente a essa sequência.
    - sequências de algarismos `L` são agrupadas num termo da forma `num(T)`, onde `T` é o átomo Prolog correspondente a essa sequência.
    - sequências de algarismos `L1`, seguidas do caracter `/`, seguidas de uma sequência de algarismos `L2` são agrupadas num termo da forma `frac(T1,T2)`, onde `T1` e `T1` são os átomos Prolog correspondentes às sequências `L1` e `L2`.
    - todos os restantes caracteres ficam inalterados, se forem caracteres aceites.

    (a) Escreva um programa que leia um ficheiro de texto, caracter a caracter, para uma lista, e construa uma lista de "tokens" usando o exercício 10. Modifique o programa para ler o ficheiro linha a linha, sendo o separador de linhas o habitual caracter de código ASCII `10` ou outro separador.

    (b) Suponha que uma receita de culinária está descrita num ficheiro com a seguinte forma:

```
Titulo: <descricao>.
Categorias: <cat1>, <cat2>, ... .
Ingredientes
   <quantidade>,<unidade>,<nome>,<descricao>;
   ...
   <quantidade>,<unidade>,<nome>,<descricao>.
Descricao: <descricao>.
```

onde `<quantidade>` é um inteiro ou uma fracção, `<descricao>` uma sequência de caracteres entre delimitadores `*` (ou outros delimitadores) e os restantes elementos entre `<` e `>` são sequências de letras.

i. Escreva um programa que leia um ficheiro da forma acima descrita, linha a linha, usando o separador de linha `.` e transforme cada linha numa lista de "tokens".

ii. Processe, com DCGs, cada uma das linhas de modo a construir uma base de dados em Prolog com predicados da forma `receita(NR,Titulo,Categorias,Descricao)` e `ingrediente(NR,Nome,Quantidade,Unidade,Descricao)` com uma cláusula para cada ingrediente. `NR` é um inteiro, que identifica a receita. Para este exercício, pode supor que `NR` é 1. `Categorias` é uma lista de termos, `Nome`, `Unidade` e `Quantidade` são termos e as restantes variáveis são sequências de caracteres.

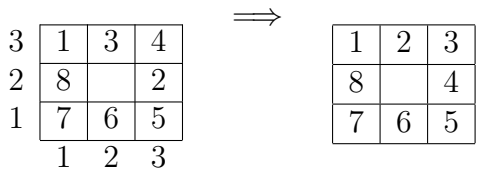
iii. Modifique o programa de modo a permitir ler um ficheiro com várias receitas da forma acima indicada.

11. **Resolução do Puzzle 8.** O Puzzle 8 é constituído por 8 quadrados numerados de 1 a 8 arranjados num tabuleiro 3x3 (9 posições). Isto significa que uma posição do tabuleiro está vazia. Em cada jogada se um quadrado estiver numa posição adjacente à posição vazia, esse quadrado pode-se mover para a posição vazia, ficando vazia a posição onde o quadrado se encontrava ("troca-se o quadrado com a posição vazia"). O puzzle diz-se resolvido se os quadrados estiverem arranhados no tabuleiro por uma ordem final pré-estabelecida. Um estado do puzzle é definido como uma lista das posições de cada quadrado da forma `X/Y`, com $1 \leq$ `X` ≤ 3, $1 \leq$ `Y` ≤ 3. O primeiro elemento indica a posição do quadrado vazio e o n-ésimo elemento a posição do quadrado $n-1$.

(a) Defina o predicado `trans(Estado1,Estado2)` que sucede se existe um movimento válido entre o estado `Estado1` e o estado `Estado2`. Comece por definir um predicado `dist` que determine a distância entre dois quadrados `Q1` e `Q2` como a soma da distância na horizontal entre `Q1` e `Q2` e a distância na vertical.

(b) Escreva um programa Prolog que resolva o puzzle dado um estado inicial e um estado final, indicando a sequência de estados que conduzem do estado inicial ao final.

Considere o estado inicial `[2/2,1/3,3/2,2/3,3/3,3/1,2/1,1/1,1/2]` e o estado final `[2/2,1/3,2/3,3/3,3/2,3/1,2/1,1/1,1/2]`, isto é,



i. Considere os métodos de procura cega
   A. em profundidade
   B. em largura

ii. Considerando a função heurística `h(Estado,V)` onde `V` é obtido calculando para cada estado duas medidas:

   `dist_total` é a soma das `distâncias` entre as posições de cada quadrado no estado `Estado` e a sua posição no estado final.

   `seq` mede o grau de ordenação dos quadrados na posição corrente em relação à ordem requerida no estado final. A medida `seq` é calculada como a soma dos *scores* de cada quadrado de acordo com as regras seguintes:
   - um quadrado no centro tem *score* `1`.
   - um quadrado não central tem *score* `0` se o quadrado é seguido, na direcção dos ponteiros do relógio, por um quadrado que é seu sucessor no estado final.
   - caso contrário o *score* do quadrado é `2`.

   O valor de `V` é dado por `dist_total+3*seq`.

   A. procura heurística subida rápida
   B. procura heurística melhor primeiro
   C. algoritmo $A^\star$

iii. Experimente com outras funções heurísticas h, por exemplo, para o caso da posição final ser,

1	2	3
4	5	6
7	8	

Exercícios de Revisão – Problemas de Exame

1. Escreva uma DCG que reconheça a linguagem $\{a^n b^n c^n \mid n \geq 0\}$

2. Escreva uma DCG que dada uma lista de caracteres transforme subsequências consecutivas de dígitos de 1 a 9 no inteiro em base 10 correspondente. Os restantes elementos da lista ficam inalterados.

3. Escreva um programa Prolog que dada uma lista X de inteiros e uma lista D de listas de inteiros retorne uma lista Z formada pelos elementos W de D que satisfaçam uma das seguintes condições:

   - W e Z diferem apenas num elemento;
   - X pode ser obtido removendo ou acrescentando um elemento a W;
   - X pode ser obtido permutando dois elementos consecutivos de W;

4. Escreva um programa Prolog que a partir de uma lista de listas L produz uma lista de listas contendo os mesmos elementos mas em que cada lista tem como primeiro elemento o seu comprimento inicial. A lista L só poderá ser visitada apenas uma vez.

5. Escreva um programa Prolog que dado um grafo não dirigido determine as suas componentes conexas. O grafo é representado por uma lista de ramos e cada ramo é representado por um par de vértices.

6. Escreva um programa Prolog que dado um tabuleiro de xadrez de lado n, determine se é possível visitar todas as casas do tabuleiro com um cavalo sem passar duas vezes pela mesma casa. Cada posição do tabuleiro pode ser representada pelo par das suas coordenadas. Se o cavalo está na posição X/Y pode mover-se para uma posição $X\pm2/Y\pm1$ ou $X\pm1/Y\pm2$ se tal posição pertencer ao tabuleiro.