Escola de Engenharia
Universidade do Minho

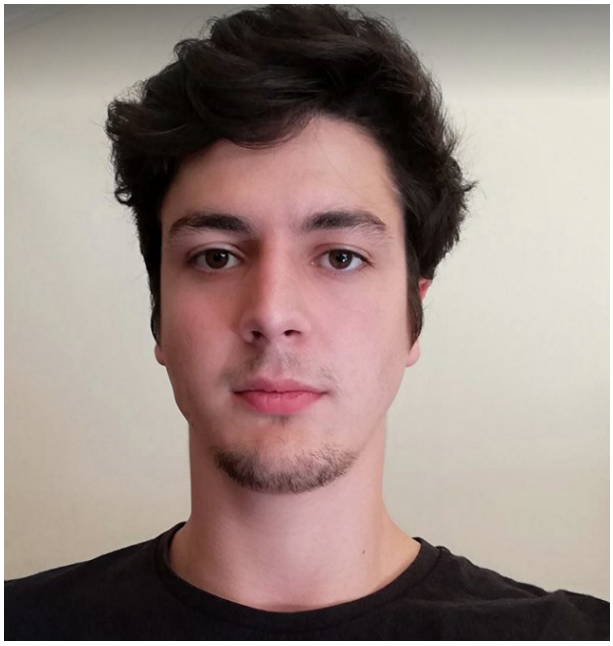
DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Mestrado Integrado em Engenharia Informática

Sistemas de Representação de Conhecimento e Raciocínio

Métodos de Resolução de Problemas e de Procura

Época Especial



Marco Gonçalves
A75480

Braga, 15 de Setembro de 2020

Conteúdo

1. Resumo

O trabalho desenvolvido pretendia abordar a programação em lógica através do PROLOG e de métodos de resolução de problemas e de procuro. Assim, era pretendido desenvolver um sistema capaz de armazenar conhecimento relativo ao metro de Washington DC e possibilitar a recomendação de trajectos a percorrer.

Ao longo deste relatório vai-se proceder à descrição do trabalho realizado bem como a sua metodologia. Assim, proceder a uma breve explicação dos algoritmos desenvolvidos, predicados usados e, de seguida, uma explicação da metodologia aplicada no desenvolvimento das funcionalidades do sistema.

Por último, será demonstrado uma breve comparação de resultados e análise destes. Em suma, acho que o trabalho desenvolvido cumpre os objectivos propostos pela equipa docente e que este possibilitou a uma melhor assimilação de conhecimentos teóricos e práticos dados pela cadeira de Sistemas de Representação de Conhecimento e Raciocínio, bem como da programação lógica em PROLOG.

2. Introdução

Este relatório é relativo ao trabalho pratico individual que substitui o teste de época especial, nesta situação pandémica que vivemos este ano, utilizando a linguagem de programação em lógica `PROLOG`, no âmbito de métodos de resolução de problemas e no desenvolvimento de algoritmos de pesquisa.

O caso de estudo na qual este trabalho se incide trata-se do metro de Washington DC. Assim é necessário ter em conta informações relativas ás estações metropolitanas e as suas ligações. Desta forma é pretendido que tenhamos um sistema capaz de representar numa base de conhecimento todos os dados relativos ao problema proposto, bem como desenvolver um sistema de recomendações de transporte para o caso de estudo.

De forma a modularizar o código guardei a base de conhecimento em ficheiros separados o ficheiro que contem os arcos (ou ligações) batizei de *arcos.pl* e o que contem as estações de *estacoes.pl* que irei explicar mais á frente e ficando assim com o ficheiro principal á qual nomeei como *metropolitan.pl*.

3. Base de Conhecimento

Um programa em Prolog é um conjunto de axiomas e de regras de inferência definindo relações entre objectos que descrevem um dado problema. A este conjunto por norma denomina-se base de conhecimentos. A base de conhecimento é essencial à representação do conhecimento e raciocínio, tendo em conta o sistema foram desenvolvidas as seguintes entidades:

3.1 Estações

```
estacao(Id, GIS_ID, Nome, Linhas, Morada, Latitude, Longitude).
```

As estações são os nossos nodos do grafo que representam os locais de origem e destino de todas as viagens. Depois do professor ter fornecido o dataset apenas acrescentei a morada de cada estação.

3.2 Ligações

```
ligacao(id1, id2, Cor).
```

Representam as ligações das respectivas linhas que existem entre as estações. Uma vez que é possível uma pessoa se deslocar em ambos os sentidos decidi criar a função **arco(id1, id2, Cor)** que representa uma ligação em ambos os sentidos mantendo a linha.

```
arco(Id1, Id2, Cor) :- ligacao(Id1, Id2, Cor).
arco(Id1, Id2, Cor) :- ligacao(Id2, Id1, Cor).
```

4. Problemas Propostos

4.1 Calcular um trajecto entre duas estações

Para a resolução deste problema usei uma pesquisa não informada em que o I representa o Inicio e F o Fim do trajecto.

```
mapa(I,F):-
        mapaAux(I,[F],R), nl,
        escreverLista(R).

mapaAux(I,[I|R1],[I|R1]).
mapaAux(I,[F|R1],R):-
        arco(X,F,_),
        \+ memberchk(X,[F|R1]),
        mapaAux(I,[X,F|R1],R).
```

4.2 Seleccionar apenas estações com uma determinada característica, para um determinado percurso

Para este problema como não adicionei características relevantes á base de conhecimento não o consegui resolver.

4.3 Excluir uma ou mais características de estações para o percurso

Assim como o problema anterior como este também depende das características eu não o consegui resolver.

4.4 Identificar quais as linhas com o maior número de possibilidades de saída num determinado percurso.

Dado um determinado percurso (lista de identificadores das estações) optei por imprimir a quantidade de cada linha que passa pelo percurso. Para isso precisei de buscar todas as cores que pertenciam a cada estação do percurso e para o conseguir recorri a um **findall**.

```
maiorSaidas(Percurso) :-
        findall(Cor,(member(X,Percurso),
                        estacao(X,_,_,Linhas,_,_,_),
                        member(Cor,Linhas)),Cores),
        escreveCores(Cores).
```

Figura 4.1: Função principal

Tendo a lista (Cores) apenas precisei de contar as ocorrências de cada uma delas e imprimir com a ajuda da função **escreveCores**.

```
escreveCores(Lista) :- nl,
        write('vermelho = '),
        count(Lista,vermelho,Vermelho),
        write(Vermelho), nl,
        write('verde = '),
        count(Lista,verde,Verde),
        write(Verde), nl,
        write('amarelo = '),
        count(Lista,amarelo,Amarelo),
        write(Amarelo), nl,
        write('azul = '),
        count(Lista,azul,Azul),
        write(Azul), nl,
        write('laranja = '),
        count(Lista,laranja,Laranja),
        write(Laranja), nl,
        write('prateado = '),
        count(Lista,prateado,Prateado),
        write(Prateado).

% count(L, X, R) Conta as as ocorrencias de X em L e guarda em R
count([],X,0).
count([X|T],X,Y):- count(T,X,Z), Y is 1+Z.
count([X1|T],X,Z):- X1\=X, count(T,X,Z).

countall(List,X,C) :-
    sort(List,List1),
    member(X,List1),
    count(List,X,C).
```

Figura 4.2: Função **escreveCores**s e **count**

```
| ?- maiorSaidas([52,53,54,51,27,25,24,23,8,6,88,89,91,9,86,59,87,2
9,28,67,71]).

vermelho = 8
verde = 1
amarelo = 1
azul = 9
laranja = 14
prateado = 9
yes
| ?-
```

Figura 4.3: Exemplo de output para o problema 4

4.5 Escolher o menor percurso

Para a resolução deste problema temos que usando o critério do menor número de estações intermédias. Para isso decidi criar a função **map2(I, F, R)** que recebendo o inicio e o fim guarda em R a rota percorrida. Tendo a rota bastou-me utilizando um **setof** que fica ordenado pelo **NEstacoes**.

```
menorPercurso(I, F) :-
        menorPercursoAux(I, F, R),
        cabeca(R, X),
        par(X,Caminho), nl,
        escreverLista(Caminho), nl,
        escreverTrajeto(Caminho).

menorPercursoAux(I, F, R) :-
        setof((NEstacoes,Caminho),
                (mapa2(I,F,Caminho),
                length(Caminho,NEstacoes)), R).
```

Figura 4.4: Problema 5

A função **cabeca(R, X)** devolve a cabeça da lista R em X. A função **par(X, Caminho)** devolve em **Caminho** o 2º elemento do par **X**.

```
| ?- menorPercurso(60,10).

60 -> 88 -> 1 -> 10
Pentagon -> L` Enfant Plaza -> Archives-Navy Mem`l -> Gallery Pl-Chinatown
yes
| ?-
| ?-
```

Figura 4.5: Exemplo de output para o problema 5

4.6 Escolher o percurso mais rápido

Neste caso para obtermos o percurso mais rápido temos usando o critério da distância. Para o resolver usei um que me devolve uma lista ordenada de pares **(Distancia, Caminho)** sendo assim temos na cabeça do **R** o par com o caminho mais rápido. A função **distancia(X, Y, D)** dados 2 i

```
maisRapido(I, F) :-
        maisRapidoAux(I, F, R),
        cabeca(R, X),
        par(X,Caminho), nl,
        escreverLista(Caminho), nl,
        escreverTrajeto(Caminho).

maisRapidoAux(I,F,R) :-
        setof((Distancia,Caminho),
                (mapa2(I,F,Caminho),
                distanciaCaminho(Caminho,Distancia)), R).
```

Figura 4.6: Problema 6

```
distanciaCaminho([X|T], Resultado) :-
        distanciaCaminho(T, X, 0, Resultado).
distanciaCaminho([], _, D, D).
distanciaCaminho([Y|T], X, Distancia, Resultado) :-
        distancia(X, Y, D),
        D2 is Distancia + D,
        distanciaCaminho(T, Y, D2, Resultado).

distancia(Id1,Id2,D) :-
        estacao(Id1,_,_,_,_,X1,Y1),
        estacao(Id2,_,_,_,_,X2,Y2),
        D is sqrt((X2-X1)^2 + (Y2-Y1)^2).
```

Figura 4.7: Função para calcular a distancia entre 2 estacoes 6

```
| ?- maisRapido(60,63).

60 -> 88 -> 89 -> 91 -> 9 -> 63
Pentagon -> L` Enfant Plaza -> Smithsonian -> Federal Triangle -> Metro Center -> McPherson Sq
yes
| ?-
```

Figura 4.8: Exemplo de output para o problema 6

4.7 Escolher o percurso que passe apenas por uma determinada linha

Na resolução deste problema usei uma abordagem Breadth First quando faço o **findall** verifico se a cor do arco pertence á lista de linhas possíveis.

```
mapaLinhasBF(I, F, Linhas) :-
        mapaLinhasBFAux([I], [], RevVisitados, F, Linhas),
        removeNaoLigados(RevVisitados, Rota), nl,
        escreverLista(Rota), nl,
        escreverTrajeto(Rota).

mapaLinhasBFAux([F|_], Historico, [F|Historico], F, Linhas).
mapaLinhasBFAux([Node|RestQ], Historico, RevVisitados, F,Linhas) :-
        findall(ProxNode, (arco(Node, ProxNode, Cor),
                           member(Cor, Linhas),
                           \+ member(ProxNode, Historico),
                           \+ member(ProxNode, RestQ)), Successors),
        append(RestQ, Successors, Queue),
        mapaLinhasBFAux(Queue, [Node|Historico], RevVisitados, F, Linhas).
```

Figura 4.9: Problema 7

```
| ?- mapaLinhasBF(88,60,[azul]).

88 -> 89 -> 91 -> 9 -> 63 -> 85 -> 79 -> 57 -> 58 -> 60
L` Enfant Plaza -> Smithsonian -> Federal Triangle -> Metro Center -> McPherson Sq -> Farragut West -> Foggy Bottom GWU -> Rosslyn -> Arlington|Cemetery -> Pentagon
yes
| ?-
```

Figura 4.10: Exemplo de output para o problema 7

4.8 Escolher uma ou mais linhas por onde o percurso deverá passar, dadas duas estações.

Dado 2 identificadores de estações imprimo todos os percursos possíveis, usando o **setof**, e quais as linhas que tem ligações entre o caminho.

```
escolherPercursos(I,F) :-
        setof(Percurso,mapa2(I,F,Percurso),Lista),
        escolherPercursosAux(Lista).
escolherPercursosAux([]).
escolherPercursosAux([H|T]) :- nl,
        escreverLista(H), nl,
        coresCaminho(H,R),
        escreveCores(R), nl,
        escolherPercursosAux(T).

coresCaminho([X|T], Resultado) :-
        coresCaminho(T, X, [], Resultado).
coresCaminho([], _, Linhas, Linhas).
coresCaminho([Y|T], X, Linhas, Resultado) :-
        setof(Cor,arco(X, Y, Cor),Cores),
        append(Linhas, Cores, NovaLinhas),
        coresCaminho(T, Y, NovaLinhas, Resultado).
```

Figura 4.11: Problema 8

```
| ?- escolherPercursos(1,88).

1 -> 10 -> 9 -> 63 -> 85 -> 79 -> 57 -> 58 -> 60 -> 88

vermelho = 1
verde = 1
amarelo = 2
azul = 6
laranja = 4
prateado = 4

1 -> 10 -> 9 -> 91 -> 89 -> 88

vermelho = 1
verde = 1
amarelo = 1
azul = 3
laranja = 3
prateado = 3

1 -> 88

vermelho = 0
verde = 1
amarelo = 1
azul = 0
laranja = 0
prateado = 0
yes
```

Figura 4.12: Exemplo de output para o problema 8

5. Conclusão

A programação em Lógica Estendida é um paradigma bastante diferente das mais comuns linguagens de programação. A linguagem de programação PROLOG é poderosa nos ramos de aprendizagem automática, processamento de linguagem natural e bases de conhecimento mas muito menos poderoso em algoritmos matemáticos ou computação gráfica. O sistema de backward chaining torna o processo é útil nos métodos de procura informada e não informada. O seu funcionamento interno torna-a mais indicada para abordagens simbólicas em oposição a abordagens baseadas em conhecimentos estatísticos.

Durante o desenvolvimento deste trabalho permitiu-me consolidar o conhecimento que é transmitido nesta unidade curricular. Foi muito gratificante ver como poderei escolher os caminhos do metro de Washington DC. Não usei uma grande variedade de algoritmos de pesquisa poderia ter implementado um A* ou mesmo um Depth First para obter resultados diferentes (tempo) mas não os testei. Como os algoritmos que aqui apresento funcionam e correctamente não senti necessidade de os utilizar.

Em suma apesar de não responder a 2 dos problemas tenho um trabalho que responde ao principal que é pedido e funciona correctamente. Tentei fazer as funções de funcionassem de forma eficiente. Tive algumas dificuldades durante a realização deste trabalho. Uma das principais dificuldades que encontrei foi a adição de novas características que apenas adicionei a morada de cada estação. Mas num futuro próximo hei de a acrescentar características e criar regras sobre elas de modo a aproximar este trabalho de um contexto mais real.