UNIVERSIDADE DO MINHO

Trabalho Individual

Mestrado Integrado em Engenharia Informática

Sistemas de Representação de Conhecimento e Raciocínio

3º Ano, 2º Semestre Ano letivo 2019/2020

A84073 Gonçalo Nuno Fernandes e Ferreira

Braga,
Maio 2020

Conteúdo

Capítulo 1

Introdução

Este trabalho individual no âmbito da unidade curricular Sistemas de Representação de Conhecimento e Raciocínio, tem como objetivo motivar o uso de programação em lógica. Tendo como finalidade a utilização de diversos tipos de conhecimento, entre eles conhecimento prefeito e imperfeito, para a construção de ferramentas que permitam solucionar o problema proposto. Como tal, é necessário um novo ponto de vista e espírito critico para o uso deste novo paradigma de programação, lógico e declarativo, introduzido na unidade curricular.

Após a cuidada analise do problema o primeiro objetivo é a representação de informação factual. Esta é desenvolvida sobre o *data set* fornecido juntamente com o enunciado, refere-se ás diversas paragens rodoviárias e adjacências entre as mesmas. Pretende-se representar de uma forma clara e objetiva esta informação crucial, pois apenas com boa informação é possível obter respostas de qualidade.

Com uma fiável base de informação passamos às heurística que permite solucionar os requisitos especificados no enunciado. A heurística visa encontrar boas soluções em maioria, num entanto não trás garantias de encontrar a melhor solução e também um processamento mais rápido. Assim levanta-se a questão de qual seria uma heurística viável e concebível, para obter resultados que se aproximem ao máximo do melhor caso, com o intuito de alcançar os requisitos mencionados no enunciado.

Capítulo 2

Base de Conhecimento

Como primeiro objetivo deste trabalho reflete-se a importação dos dados fornecidos, estes estão no formato *".xlsx"* que é possível ser visualizado em folhas de calculo como por exemplo o utensílio usado *LibreOffice Calc*. Os dados são compostos por dois ficheiros paragens e adjacências, com a informação total de cada paragem e informação de paragens correspondentes a uma carreira respetivamente. Para facilitar a manipulação destes usou-se um *script* para exportar todas as *sheets* pertencentes para o formato *csv (Comma Separated Values)*, deste modo a representa-los de um modo mais genérico. Tendo todos os ficheiros CSV pretendidos optou-se escrever dois programas em *python*, para paragens e adjacências respetivamente, devido à diferente forma de representar a base de conhecimento. Os factos em prolog são definidos no output do *parser* em *python* no qual é tratado os diversos tipos de faltas de informação. Deste modo é necessário representar conhecimento perfeito e imperfeito, tendo em conta as inconsistências do *data set* fornecido.

2.1 Paragens

As paragens são representadas na base de conhecimento como "paragem"tendo como conteúdo toda os dados úteis respetivos, isto permite uma fácil manipulação de dados. Os dados representados são 11 campos contendo: identificador único, latitude, longitude, estado de conservação, tipo de abrigo, abrigo com publicidade, operadora, carreiras, código de rua, nome de rua e freguesia.

2.2 Adjacências

Para representar as adjacências como conhecimento que permite especificar a conexão entre paragens são usados 3 campos que representando os identificadores únicos das paragens que têm adjacência e a carreira a que pertence. Esta representação de conhecimento indica que adjacências existem, assim fazendo um grafo representativo de todas as carreiras e paragens existentes.

Capítulo 3

Proposta de Solução

A solução proposta visa alcançar todos os requisitos propostos, mais ainda que o *software* produzido fosse genérico assim possibilitando a atualização e viabilidade da solução. Assim foi divido em dois grupos Heurística e Requisitos.

3.1 Heurística

O grupo Heurística tem como objetivo explicar o algoritmo utilizado em cada um dos cenários abaixo representados, sendo que este deve contemplar uma forma adequada de resolver cada problema.

- **Calcular um trajeto entre dois pontos**

  Com objetivo de calcular o trajeto entre dois pontos sem restrições adicionais é utilizado um simples algoritmo de *Depth First* para obter o resultado, apenas garante-se que este algoritmo não permita visitar paragens anteriormente visitadas.

- **Escolher o menor percurso (usando critério menor número de paragens)**

  Para a resolução deste problema é proposto o uso do algoritmo *Breadth First*, pois as suas características favorecem a resposta pretendida. Isto é, o algoritmo começa por expandir para todos os nodos filhos e categorizando-os como visitados, assim não repetindo cálculos desnecessários, ao encontrar a paragem pretendida retorna o caminho percorrido.

- **Escolher o percurso mais rápido (usando critério da distância)**

  No âmbito de obter o caminho mais perto tendo em conta a distancia entre os pontos, foi determinada uma heurística que tem como objetivo garantir que o próxima paragem seja a mais perto do destino. Para atingir este objetivo foi feito o predicado *check_closer_adjacencia* para conter a lógica por de trás de está decisão, que se baseá no calculo de distancias entre paragens e a sua comparação.

```
% - Checks a path from B to A - %

check_path(A, B, P) :-
    build_path(A, [B], P)
.

build_path(A, [A|P1], [A|P1]).
build_path(A, [Y|P1], P) :-
   check_closer_adjacencia(X, Y, [Y|P1]),
   build_path(A, [X,Y|P1], P)
.
```

Figura 3.1: Construção de um trajeto.

3.2 Requisitos

O grupo Requisitos engloba os objetivos que são dependentes de existir o algoritmo que construa um caminho ou mesmo um caminho. Estes têm uma mais fácil implementação pois apenas aplicam uma filtragem ao trajeto a seguir.

- **Selecionar apenas algumas das operadoras de transporte para um determinado percurso**

  De modo ao trajeto apenas ter paragens de um grupo de operadoras inseriu-se o predicado indicado na imagem na validação da próxima paragem ao construir o caminho.

```
% - Next stop from a set of firms - %

prox_in_setOfFirm(A, B, SET) :-
    adjacencia(A, B, _),
    paragem(A,_,_,_,_,_,A_FIRM,_,_,_,_),
    paragem(B,_,_,_,_,_,B_FIRM,_,_,_,_),
    memberchk(A_FIRM, SET),
    memberchk(B_FIRM, SET)
.
```

Figura 3.2: Próxima paragem pertence a operadoras.

- **Excluir um ou mais operadores de transporte para o percurso**

  Analogamente á alínea anteriormente utiliza-se o predicado abaixo para restringir a próxima paragem de modo a que não pertença a um grupo de operadoras.

```
% - Next stop not from a set of firms - %

-prox_in_setOfFirm(A, B, SET) :-
    adjacencia(A, B, _),
    paragem(A,_,_,_,_,_,A_FIRM,_,_,_,_),
    paragem(B,_,_,_,_,_,B_FIRM,_,_,_,_),
    \+ memberchk(A_FIRM, SET),
    \+ memberchk(B_FIRM, SET)
.
```

Figura 3.3: Próxima paragem não pertence a operadoras.

- **Identificar quais as paragens com o maior número de carreiras num determinado percurso**

  Dado um caminho retorna a lista de *gids* das paragens que têm mais carreiras que passam por ela. Para obter este resultado fez-se um *map* sobre a lista de paragens colocando esse mesmo valor como *Key*, assim possibilitando a ordenação por esse critério. Invertendo a ordem dessa lista obtém-se as paragens com mais carreiras.

```
% - Returns the stops with most careers - %

max_carreiras(PATH, LIST_WITH_MORE_CARR) :-
    maplist(aux_num_carreiras, PATH, NUM_CARR),
    keys_and_values(KV_LIST, NUM_CARR, PATH),
    keysort(KV_LIST, SORTED),
    reverse(SORTED, [H|T]),
    check_equal_key(H, T, KV_MAX),
    keys_and_values(KV_MAX, _, LIST_WITH_MORE_CARR)
.
```

Figura 3.4: Paragem com mais carreiras.

- **Escolher o percurso que passe apenas por abrigos com publicidade**

  Restringindo a próxima paragem a ter publicidade utiliza-se o predicado abaixo na procura de um caminho.

```
% - Next stop with advertisement - %

prox_with_advertisement(A, B) :-
    adjacencia(A, B, _),
    paragem(A,_,_,_,_,A_ADV,_,_,_,_,_),
    paragem(B,_,_,_,_,B_ADV,_,_,_,_,_),
    A_ADV == 'Yes',
    B_ADV == 'Yes'
.
```

Figura 3.5: Próxima paragem com publicidade.

- **Escolher o percurso que passe apenas por paragens abrigadas**

  De modo a filtrar as próximas paragens apenas permitindo que sejam as que tenham abrigo usa-se o predicado abaixo.

```
% - Next stop with cover - %

prox_with_cover(A, B) :-
    adjacencia(A, B, _),
    paragem(A,_,_,_,A_COVER,_,_,_,_,_),
    paragem(B,_,_,_,B_COVER,_,_,_,_,_),
    \+ A_COVER == 'Sem Abrigo',
    \+ B_COVER == 'Sem Abrigo'
.
```

Figura 3.6: Próxima paragem com abrigo.

- **Escolher um ou mais pontos intermédios por onde o percurso deverá passar**

  Para completar o requisito de fornecer pontos intermédios para um dado caminho, assumiu-se que estes pontos são dados na ordem que o utilizador pretende passar. Deste modo a solução proposta sugere o calculo dos caminhos entre os pontos dados pela dada ordem, concatenando estes.

```
% - Checks a path from B to A with - %
% - the intermidiate points given. - %

check_path_inter(A, B, L, R) :-
    append([L,[B]|[]], LR),
    build_path_inter(A, LR, P),
    append(P, R)
.

build_path_inter(_, [], []).
build_path_inter(A, [H|T], [NP|P]) :-
    check_path(A,H,NP),
    build_path_inter(H, T, P)
.
```

Figura 3.7: Caminho com paragens intermédias.

Capítulo 4

Conclusão

Como este trabalho individual foi possível aplicar a matéria lecionada na unidade curricular, desde a criação de uma base de conhecimento até a evolução do conhecimento. Com os requisitos propostos no enunciado observou-se a importância de ter uma boa base de conhecimento, desde os dados *raw* até tratados e importados, causando um impacto significativo nas funcionalidades pretendidas. A evolução do conhecimento é dependente da qualidade de da base conhecimento, num entanto a grande dificuldade surge na definição das heurísticas para a resolução dos problemas, estas devem ser desenhadas com o propósito á imagem do problema que visa resolver. Em suma este trabalho proporcionou a familiarização com programação lógica e as suas vantagens, como também equacionar resoluções para problemas não determinísticos.

Capítulo 5

Referencias

- **Sicstus:**
  https://sicstus.sics.se/sicstus/docs/latest4/html/sicstus.html/index.html
- **Sicstus library(sets):**
  https://sicstus.sics.se/sicstus/docs/latest4/html/sicstus.html/lib_002dsets.html
- **Sicstus library(lists):**
  https://sicstus.sics.se/sicstus/docs/4.3.0/html/sicstus/lib_002dlists.html
- **Breadth First:**
  https://ksvi.mff.cuni.cz/~dingle/2019-20/npp/notes_5.html