Relatório da Unidade Curricular de Inteligência Artificial 2021/2022

Grupo 09
Bernardo Saraiva 93189
José Gonçalves 93204
Pedro Araújo 70699
Rui Moreira 93232

Dezembro, 2021

Índice

1 Introdução

O presente documento tem o intuito de explicar os procedimentos adoptados para a realização da primeira fase do Instrumento de Avaliação em Grupo. Esta fase passou pela criação de um sistema de representação de conhecimento e raciocínio capaz de dar resposta a um problema de universo de discurso na área da logística e distribuição de encomendas.

2 Base de Conhecimentos

A presente Secção 2 tem o intuito de estabelecer a base de conhecimento que suporta as *queries* que serão apresentadas e discutidas na secção seguinte (Secção 3). Temos então o conhecimento representado do seguinte modo:

- *Veiculo*: O conhecimento passa pela atribuição de características predefinidas para cada um dos diferentes tipos de veículos à disposição da empresa. Assim sendo, um veículo detém as seguintes características,:
  - o tipo - Bicicleta, Mota e Carro;
  - a sua velocidade média;
  - a carga máxima que este é capaz de transportar;
  - a sua posição ecológica face aos restantes tipos de veículo*.
- *Estafeta*: O conhecimento é representado:
  - pela identificação do estafeta em questão;
  - pela zona onde se encontra alocado para realizar as suas entregas;
  - por qual dos veículos possui para realização das mesmas.
- *Encomenda*: O conhecimento é denominado pela/o:
  - identificação da encomenda;
  - identificação do estafeta responsável pela mesma;
  - identificação do cliente à qual a mesma se destina;
  - zona habitacional onde será entregue;
  - data de previsão de entrega;
  - classificação associada ao estafeta que efectuou a entrega;
  - sua volumetria;
  - seu custo monetário.

A posição ecológica foi determinada seguindo a seguinte lógica:* ***Bicicleta* *>* ***Mota*** *>* ***Carro****, sendo a bicicleta o veículo mais ecológico e o carro o menos ecológico.*

Podemos observar na Figura 1, as estruturas acima descritas já povoadas com dados.

```
% veiculo (tipo de veiculo, velocidade_media, cargaMax, vertente_ecologica)
veiculo(bicicleta, 10, 5, 50).
veiculo(mota, 35, 20, 30).
veiculo(carro, 25, 100, 10).

% estafeta(idEstafeta, freguesia, veiculo)
estafeta(joaquim, vila-caiz, carro).
estafeta(painatal, roriz, carro).
estafeta(rui, roriz, mota).
estafeta(ze-joao, barcelos, bicicleta).
estafeta(miguel, braga, bicicleta).
estafeta(margarida, guimaraes, mota).
estafeta(gigachad, braga, bicicleta).

% encomenda(idEncomenda, idEstafeta, idCliente, freguesia/rua , dataMax/dataEntrega , classificação, peso/volume, preço)
encomenda(televisao, joaquim, manuel, vila-caiz/aldeia-nova, data(2021, 1, 30)/data(2021, 1, 29),5, 30/80, 10).
encomenda(portatil, rui, bernardo, roriz/pidre, data(2021, 2, 12)/data(2021, 2, 11),4 , 12/30, 5).
encomenda(telemovel, ze-joao, miguel, barcelos/pedreira, data(2021, 3, 10)/data(2021, 1, 29), 3, 3/10, 3).

encomenda(forno, painatal, bernardo, roriz/pidre, data(2021, 2, 12)/data(2021, 3, 29), 1, 12/30, 5).
encomenda(telemovel, margarida, pedro, vila-caiz/aldeia-nova, data(2021, 3, 10)/data(2021, 3, 9), 2, 3/10, 3).
encomenda(teclado, rui, alberto, esposende/margem, data(2021, 6, 19)/data(2021, 5, 22), 5, 21/30, 4).
encomenda(rato, miguel, joao, esposende/margem, data(2021, 6, 22)/data(2021, 6, 22), 5, 2/30, 50).
encomenda(headset, margarida, ana, esposende/margem, data(2021, 12, 30)/data(2021, 12, 29), 3, 9/30, 24).

encomenda(pao, gigachad, ana, braga/vila-verde,data(2021, 3, 10)/data(2021, 1, 29), 3, 3/10, 3).
encomenda(pizza, gigachad, bernardo, braga/vila-verde,data(2021, 3, 11)/data(2021, 1, 30), 3, 3/10, 3).
encomenda(hamburger, gigachad, antonio, braga/real, data(2021, 3, 30)/data(2021,3,15), 5 , 4/11 , 5).
```

Figure 1: Base de conhecimento povoada com dados

3 Funcionalidades

Nesta Secção 3, serão apresentadas e discutidas as diferentes *queries* propostas a serem solucionadas.

3.1 Comparação de datas

De modo a ser possível efectuar uma comparação de datas eficiente foi elaborado o predicado apresentado de seguida:

```
compare_data(data(YY,MM,DD), = ,data(YY,MM,DD)).

compare_data(data(Y,M,D), > ,data(YY,MM,DD)) :-
        Y > YY.
compare_data(data(Y,M,D), > ,data(Y,MM,DD)) :-
        M > MM.
compare_data(data(Y,M,D), > ,data(Y,M,DD)) :-
        D > DD.

compare_data(data(Y,M,D), < ,data(YY,MM,DD)) :-
        Y < YY.
```

```
compare_data(data(Y,M,D), < ,data(Y,MM,DD)) :-
        M < MM.

compare_data(data(Y,M,D), < ,data(Y,M,DD)) :-
        D < DD.
```

Este possui aridade 3, requerendo por isso duas datas e um sinal para que seja possível realizar a comparação. O raciocínio deste predicado consiste em comparar os elementos da data por ordem decrescente de prioridade, começando assim pelo ano, mês e por fim o dia.

3.2 *Query 1*

Pretende-se calcular o estafeta que utilizou mais vezes o meio de transporte mais ecológico. Para atingir este objetivo, foi criado o predicado *maisEcologico* bem como vários predicado auxiliares, como é possível observar no código transcrito.

```
% Coloca no res todos os ids de estafeta
% que usaram determinado veículo

findEstafetasPorVeiculo(Veiculo, Res) :-
        findall(IdEstafeta, estafeta(IdEstafeta,
        _,Veiculo), Res).


calculaEncomendasEstafeta(IdEstafeta,Res) :-
        findall(IdEncomenda,encomenda(IdEncomenda,
        IdEstafeta, _, _, _, _, _, _),Lista), length(Lista,Res).


descending([], []).
descending([A], [A]).
descending(A,  [X,Y|C]) :-
  select(X, A, B),
  descending(B, [Y|C]),
        calculaEncomendasEstafeta(X,W),
        calculaEncomendasEstafeta(Y,Z),
          W   >=    Z.

iguais([X|[]],1).

iguais([X,Y|Xs], 1) :-  calculaEncomendasEstafeta(X,Z),
                        calculaEncomendasEstafeta(Y,W),
                        Z \= W.
```

```
iguais([X,Y|Xs],Res) :- calculaEncomendasEstafeta(X,Z),
                        calculaEncomendasEstafeta(Y,W),
                        Z = W,
                        iguais([Y|Xs],Res2),
                        Res is 1 + Res2.

maisEcologico(Lista,Res):-
        iguais(Lista,Iguais),
        take(Iguais,Lista,Res).

encontraMaisEcologico(Res) :-
        findEstafetasPorVeiculo(bicicleta,ListaEstafetasBicicleta),
        length(ListaEstafetasBicicleta,L),
        L > 0,
        descending(ListaEstafetasBicicleta,Lista),
        maisEcologico(Lista,Res),
        !.

encontraMaisEcologico(Res) :-
        findEstafetasPorVeiculo(mota,ListaEstafetasMota),
        length(ListaEstafetasMota,L),
        L > 0,
        descending(ListaEstafetasMota,Lista),
        maisEcologico(Lista,Res),
        !.

encontraMaisEcologico(Res) :-
        findEstafetasPorVeiculo(carro,ListaEstafetasCarro),
        length(ListaEstafetasCarro,L),
        L > 0,
        descending(ListaEstafetasCarro,Lista),
        maisEcologico(Lista,Res),
        !.
```

Inicialmente, cria-se uma lista com todos os estafetas que usam o veículo mais ecológico existente na base de conhecimento, sendo este a bicicleta. De seguida, ordena-se a lista com os IDs do estafeta por ordem decrescente de entregas, retirando depois os elementos com o mesmo número de entregas da cabeça, uma vez que podem existir estafetas com o mesmo grau de ecologia.

3.3 *Query 2*

De modo a identificar o estafetas que efectuaram um determinado número de entregas a um cliente, criou-se o predicado *encomendas_do_estafeta_PorCliente* com aridade 3. Este recebe o Id do cliente, quais as encomendas entregues por

um determinado estafeta, a lista actual dos estafetas (inicialmente vazia) e a variável onde será devolvida a resposta.

```
encomendas_do_estafeta_PorCliente(_, [], Lista, Lista).

encomendas_do_estafeta_PorCliente(IdCliente, [IdEncomenda|Xs], Lista, Res) :-
        findall(IdEstafeta, encomenda(IdEncomenda,
        IdEstafeta, IdCliente, _, _ , _, _, _), S),
        append(Lista, S, L),
        encomendas_do_estafeta_PorCliente(IdCliente, Xs, L, Res).
```

Primeiramente, foi elaborado o caso de paragem. Caso a lista das encomendas se encontre vazia, é devolvida a lista actual das encomendas - 3º argumento.

Caso contrário, procuram-se todos os estafetas que efectuaram a primeira encomenda presente na lista e de seguida esta informação é alocada numa lista auxiliar. Obtendo-se esta lista, é de seguida adicionada à lista actual dos estafetas - 3º argumento - repetindo-se recursivamente o processo para as restantes encomendas às quais se pretende conhecer o estafeta responsável pela entrega. Eventualmente será alcançado o caso de paragem, terminando assim a execução.

3.4 *Query 3*

Com o intuito de identificar os clientes servidos por um determinado estafeta, tal como proposto, elaborou-se o predicado *clientesPorEstafeta*.

```
clientesPorEstafeta(IdEstafeta, Res) :-
        findall(IdCliente, encomenda(_,
        IdEstafeta, IdCliente, _, _, _, _), Res).
```

Neste predicado, de aridade 2, é primeiramente passado o Id do estafeta e de seguida a variável de retorno. É realizado um *findall* de todos os clientes associados ao referido estafeta colocando-se, por fim, na variável de resposta que será devolvida como lista.

3.5 *Query 4*

De modo a calcular o valor facturado pela ***Green Distribution*** num determinado dia, foi criada uma lista com os preços de todas as entregas referentes a um determinado dia - recorrendo ao predicado *findall* - deste modo apenas foi necessário efectuar o somatório de todos os elementos da referida lista.

```
somaElementos([], 0).
somaElementos([H|Xs], Res) :-
        somaElementos(Xs, Sum),
        Res is Sum+H.

faturacaoDiaria(DataEntrega, Res) :-
```

```
        findall(Preco, encomenda(_, _, _, _,
        _/DataEntrega , _, _, Preco), S),
        somaElementos(S, Res).
```

3.6 *Query 5*

Com o objectivo de identificar quais as zonas, rua ou freguesia, com maior volume de entregas pela ***Green Distribution***, efectuou-se o seguinte processo, representado pela seguinte transcrição:

```
listaDasZonas(Res) :-
        findall(Zona, encomenda(_, _, _,
        Zona, _, _, _, _), S), sort(S, Res).

entregasPorZona(Zona, Res) :-
        findall(Zona, encomenda(_, _, _,
        Zona, _, _, _, _), S),
        length(S, Res).

entregasPorZonaLista([], Lista, Lista).
entregasPorZonaLista([Zona|Xs], Lista, Res) :-
        entregasPorZona(Zona, X1),
        append(Lista, [Zona/X1], L3),
        entregasPorZonaLista(Xs, L3, Res).

zonasComMaisEntregas(Res) :-
    listaDasZonas(ListaZonas),
    entregasPorZonaLista(ListaZonas, [], Lista),
    pair_sort(Lista, Res).

%Auxiliar que ordena a lista de tuplos
:-use_module(library(clpfd)).

swap_internals((X/Y), Y1-X):- Y1 #= -Y.

pair_sort(L,Sorted):-
      maplist(swap_internals, L, L2),
      keysort(L2, L3),
      maplist(swap_internals, Sorted, L3).
```

Primeiramente foi criada uma lista com todas as zonas, recorrendo ao uso do predicado auxiliar *listaDasZonas*.

De seguida, para cada zona verificou-se quantas entregas foram efectuadas, recorrendo ao predicado auxiliar *entregasPorZona*, criando-se uma lista com o tuplo *Zona/Número de Entregas*.

Por fim, efectuou-se a ordenação da lista recorrendo ao segundo elemento de cada tuplo. Deste modo, foi possível obter a lista ordenada das zonas tendo em conta o número de entregas efectuadas na mesma.

3.7 *Query 6*

De modo a calcular a classificação média de satisfação de um cliente face a um determinado estafeta é por criada uma lista com todas as classificações associadas a um estafeta, recorrendo ao uso do predicado *findall*. De seguida. efectuou-se a divisão do somatório de todos os valores listados pelo número de elementos presentes na lista, obtendo-se assim a classificação média.

```
somaLista([],0).
somaLista([H|T],S) :-
        somaLista(T,G),
        S is H+G.

classificacaoDoClienteParaEstafeta(IdEstafeta,Res) :-
        findall(Classificacao,encomenda(_,
        IdEstafeta, _, _, _, Classificacao, _, _), Lista),
        somaLista(Lista,S),
        length(Lista,T),
        Res is S / T.
```

3.8 *Query 7*

Com o intuito de identificar o número total de entregas executadas pelos diferentes meios de transporte num determinado intervalo de tempo, criou-se uma lista de um tuplos, contendo *Data/IdEstafeta* de todas as entregas efectuadas no intervalo de tempo pretendido. Após a obtenção dos ids de estafetas que efectuaram entregas no intervalo de tempo pretendido, é criada uma lista com todos os veículos usados nas mesma, uma vez que a cada estafeta está associado um veículo. De modo a descobrir o número de entregas efectuadas por cada veículo foram criados três predicados auxiliares com a finalidade contar quantas vezes cada tipo de veículo aparece nesta lista, sendo estes valores devolvidos. Desmontra-se em seguida o código relativo aos predicados correspondentes:

```
listaEntregasDurante(DataI/DataF,CL) :-
        findall(Data/IdEstafeta,encomenda(_ ,
        IdEstafeta, _, _, _/Data, _, _, _), L),
        removeListaEntregasForaDoIntervalo(DataI/DataF, L, CL).


removeListaEntregasForaDoIntervalo(_, [], []).

removeListaEntregasForaDoIntervalo(DataI/DataF, [X/IdEstafeta|XS], Res):-
```

```
        compare_data(X, >, DataI),
        compare_data(X, <, DataF),
        removeListaEntregasForaDoIntervalo(DataI/DataF, XS, Y),
        append([X/IdEstafeta], Y, Res).

 removeListaEntregasForaDoIntervalo(DataI/DataF, [X|XS], Res) :-
        removeListaEntregasForaDoIntervalo(DataI/DataF, XS, Res).

criaListaVeiculo([],Lista,Lista).
criaListaVeiculo([_/Id|Estafetas],Lista, Res) :-
        findall(Veiculo,estafeta(Id,_,Veiculo),V),
        append(Lista,V,L8),
        criaListaVeiculo(Estafetas,L8,Res).


contaBicicletas([],0).
contaBicicletas([bicicleta|T],N) :-
        contaBicicletas(T,N1), N is N1 + 1.
contaBicicletas([X|T],N) :-
        X \= bicicleta,
        contaBicicletas(T,N).

contaCarros([],0).
contaCarros([carro|T],N) :-
        contaCarros(T,N1), N is N1 + 1.
contaCarros([X|T],N) :-
        X \= carro,
        contaCarros(T,N).

contaMotas([],0).
contaMotas([mota|T],N) :-
        contaMotas(T,N1), N is N1 + 1.
contaMotas([X|T],N) :-
        X \= mota,
        contaMotas(T,N).

numeroTotalEntregas(DataI/DataF,EntregasBicicleta,
        EntregasCarro,EntregasMoto) :-
        listaEntregasDurante(DataI/DataF,ListaEstafetas),
        criaListaVeiculo(ListaEstafetas,[],ListaVeiculos),
        contaBicicletas(ListaVeiculos,EntregasBicicleta),
        contaCarros(ListaVeiculos,EntregasCarro),
        contaMotas(ListaVeiculos,EntregasMoto),
        !.
```

3.9 *Query 8*

Ao contrário da *Query 7*, presente na Subsecção3.8, o propósito é agora identificar o número total de entregas efectuadas por estafetas num determinado intervalo de tempo.

Deste modo, foi criada uma lista com todas as entregas, recorrendo novamente ao predicado *findall*), de onde foram excluídas todas as entregas fora do período temporal a ser contemplado (com recurso ao predicado auxiliar *compare_ data*, que pode ser encontrado no início). Por fim, foram efectuados cálculos sobre o tamanho da lista com o intuito de se ficar a conhecer o número total de entregas efectuadas no período de tempo contemplado.

```
entregasDurante(DataI/DataF,Res) :-
        findall(Data,encomenda(_, _, _, _, _/Data, _, _, _), L),
        removeEntregasForaDoIntervalo(DataI/DataF, L, CL),
        length(CL,Res).

removeEntregasForaDoIntervalo(_, [], []).

removeEntregasForaDoIntervalo(DataI/DataF, [X|XS], Res):-
        compare_data(X, >, DataI),
        compare_data(X, <, DataF),
        removeEntregasForaDoIntervalo(DataI/DataF, XS, Y),
        append([X], Y, Res).

removeEntregasForaDoIntervalo(DataI/DataF, [X|XS], Res) :-
        removeEntregasForaDoIntervalo(DataI/DataF, XS, Res).
```

3.10 *Query 9*

Com o intuito de calcular o número de encomendas entregues e não entregues pela empresa ***Green Distribution*** num intervalo de tempo, desenvolveu-se o predicado *calculaEncomandasIntervalo* com aridade 3.

Primeiramente, este recebe um tuplo com as datas que marcam o início e o fim do intervalo de temporal em questão. De seguida, são passadas as variáveis que irão guardar o número de encomendas entregues e o número de encomendas não entregues, sendo estas as respostas ao problema.

De modo a obter o resultado proposto é calculado o total de encomendas efectuadas, mais uma vez recorrendo ao predicado *findall*, sendo as mesmas guardadas numa lista com os respectivos id's. Fazendo uso do predicado *length*, é facultado o calculo do tamanho da lista em questão.

Com o auxilio do predicado desenvolvido para a *Query 8*, presente na Subsecção 3.9, que devolve o número total de entregas no intervalo pretendido, coloca-se este valor numa das variáveis de resposta.

Finalmente, é possível calcular o número de encomendas não entregues. Para tal, este valor é subtraído ao número total de encomendas entregues. De modo a que este predicado não recorra ao mecanismo de *backtracking*, é adicionou-se um

cut ("!"), fazendo com que seja devolvido o resultado pretendido e possibilitando o término.

```
calculaNEncomendasIntervalo(DataI/DataF,
        ResEntregues, ResNaoEntregues) :-
        findall(IdEncomenda,encomenda(IdEncomenda,
        _, _, _, _, _, _, _), L),
        length(L, TotalEncomendas),
        entregasDurante(DataI/DataF, ResEntregues),
        ResNaoEntregues is (TotalEncomendas - ResEntregues),
        !.
```

3.11 *Query 10*

Com o intuito de calcular o peso total transportado por um estafeta num determinado dia, efectuou-se a criação de uma lista com todos os estafetas do sistema.

Posteriormente, para cada estafeta, calculou-se o peso total que ele entregou num determinado dia, recorrendo ao predicado auxiliar *pesoTotalPorEstafeta*).

```
pesoTotalPorEstafeta(IdEstafeta,PT) :-
        findall(Peso, encomenda(_, IdEstafeta,
        _, _, _/DataEntrega , _, Peso/_ , _), S),
        somaElementos(S, PT).

pesoTotalPorEstafetasLista([], Lista, Lista).
pesoTotalPorEstafetasLista([Estafeta|Estafetas],Lista,Res) :-
        pesoTotalPorEstafeta(Estafeta,PT),
        append(Lista, [Estafeta/PT], L10),
        pesoTotalPorEstafetasLista(Estafetas,L10,Res).


pesoTotalPorEstafetas(Res) :-
        findall(IdEstafeta, estafeta(IdEstafeta, _, _),ListaEstafetas),
        pesoTotalPorEstafetasLista(ListaEstafetas,[],Res).
```

4 Testes Computacionais

Na presente Secção 4, pretende-se demonstrar a aplicação prática de toda a estrutura e predicados discutidos nas secções e subsecções anteriores, demonstrando-se os testes computacionais efetuados para cada query e os respetivos resultados obtidos, sempre recorrendo à base de dados que se encontra na *Figura1* da *Secção2*.

4.1 *Query 1*

```
?- encontraMaisEcologico(Res).
Res = [gigachad].
```

Figure 2: Resultados obtidos para a *Query 1*

4.2 *Query 2*

```
?- encomendas_do_estafeta_PorCliente(ana,[pao,headset],[],Res).
Res = [gigachad, margarida].
```

Figure 3: Resultados obtidos para a *Query 2*

4.3 *Query 3*

```
?- clientesPorEstafeta(gigachad,Res).
Res = [ana, bernardo, antonio].
```

Figure 4: Resultados obtidos para a *Query 3*

4.4 *Query 4*

```
?- faturacaoDiaria(data(2021,1,29), Res).
Res = 16.
```

Figure 5: Resultados obtidos para a *Query 4*

4.5 *Query 5*

```
?- zonasComMaisEntregas(Res).
Res = [esposende/margem/3,  (vila-caiz/aldeia-nova)/2,  (braga/vila-verde)/2, roriz/pi
dre/2, barcelos/pedreira/1, braga/real/1].
```

Figure 6: Resultados obtidos para a *Query 5*

4.6 *Query 6*

```
?- classificacaoDoClienteParaEstafeta(gigachad,Res).
Res = 3.6666666666666665.
```

Figure 7: Resultados obtidos para a *Query 6*

4.7 *Query 7*

```
?- numeroTotalEntregas(data(2020,1,1)/data(2022,12,30),EntregasBicicleta,EntregasCarro
,EntregasMoto).
EntregasBicicleta = 5,
EntregasCarro = 2,
EntregasMoto = 4.
```

Figure 8: Resultados obtidos para a *Query 7*

4.8 *Query 8*

```
?- entregasDurante(data(2020,1,1)/data(2022,12,30), Res).
Res = 11.
```

Figure 9: Resultados obtidos para a *Query 8*

4.9 *Query 9*

```
?- calculaNEncomendasIntervalo(data(2021,1,1)/data(2021,12,19), Entregues, NaoEntregue
s).
Entregues = 10,
NaoEntregues = 1.
```

Figure 10: Resultados obtidos para a *Query 9*

4.10 *Query 10*

```
?- pesoTotalPorEstafetas(Res).
Res = [joaquim/30, painatal/12, rui/33,  (ze-joao)/3, miguel/2, margarida/12, gigachad
/10].
```

Figure 11: Resultados obtidos para a *Query 10*

5 Conclusões

A implementação prática do problema proposto, que retrata um caso de contexto real de uma empresa de logística e distribuição de encomendas, permitiu a exploração e consolidação dos conhecimentos até agora adquiridos sobre a linguagem de programação *PROLOG*. Deste modo, foi possível entender e reconhecer as potencialidades e capacidades da linguagem aquando da sua aplicação a casos reais. O sistema criado é capaz de inferir conhecimento relativo a uma panóplia de informações, como foi possível visualizar nas secções e subsecções anteriores. Apesar da base de conhecimento não ser muito extensa, a mesma encontra-se capaz de dar resposta a todas as *queries* de forma eficiente.

References

[1] I. Bratko, *Prolog Programming for Artificial Intelligence*. International Computer Science Series, Addison-Wesley, 2011.

[2] S. Russell, S. Russell, and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson series in artificial intelligence, Pearson, 2020.