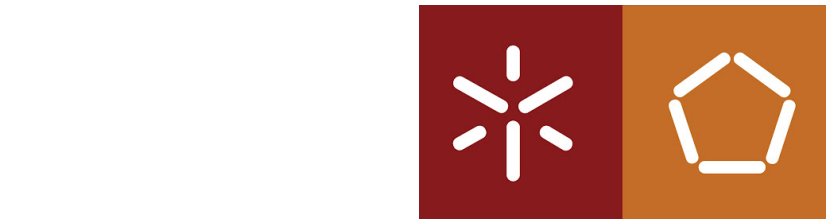

Universidade do Minho

Licenciatura em Engenharia Informática

Inteligência Artificial

Trabalho em grupo – 1ª Fase

Grupo 41

Rui Monteiro (A93179) Rodrigo Rodrigues (A93201)

Daniel Azevedo (A93324) Nuno Peixoto (A93244)

Ano Letivo 2021/2022

1 Resumo

Este documento é um relatório técnico sobre o trabalho desenvolvido durante a primeira fase do trabalho prático da Unidade Curricular Inteligência Artificial. A execução do exercício privilegia os predicados que melhor favorecem clareza e simplicidade, fazendo bom uso da linguagem de programação em lógica *PROLOG*. Todo o código aqui apresentado está na íntegra no ficheiro em anexo.

Conteúdo

1 Resumo **2**

2 Introdução **4**

3 Preliminares **5**

4 Descrição do Trabalho e Implementação **6**
4.1 Caracterização de Conhecimento 6
4.2 Predicados gerais/auxiliares 7
4.3 Base de Conhecimento inicial 10
4.4 Cálculo dos Custos de Entrega 11

5 Funcionalidades do Programa **13**
5.1 Funcionalidade 1: identificar o estafeta que utilizou mais vezes um meio de transporte mais ecológico 14
5.2 Funcionalidade 2: identificar que estafetas entregaram determinada(s) encomenda(s) a um determinado cliente 15
5.3 Funcionalidade 3: identificar os clientes servidos por um determinado estafeta . . . 15
5.4 Funcionalidade 4: calcular o valor faturado pela Green Distribution num determinado dia 15
5.5 Funcionalidade 5: identificar quais as zonas (e.g., rua ou freguesia) com maior volume de entregas por parte da Green Distribution 16
5.6 Funcionalidade 6: calcular a classificação média de satisfação de cliente para um determinado estafeta 17
5.7 Funcionalidade 7: identificar o número total de entregas pelos diferentes meios de transporte, num determinado intervalo de tempo 17
5.8 Funcionalidade 8: identificar o número total de entregas pelos estafetas, num determinado intervalo de tempo 18
5.9 Funcionalidade 9: calcular o número de encomendas entregues e não entregues pela Green Distribution, num determinado período de tempo 19
5.10 Funcionalidade 10: calcular o peso total transportado por estafeta num determinado dia 19
5.11 Exemplos de Resultados das Funcionalidades 20

6 Conclusão **21**

2 Introdução

Neste documento apresentamos uma solução para a primeira fase do trabalho prático da Unidade Curricular de Inteligência Artificial, perante a proposta de realizar um sistema de representação de conhecimento e raciocínio com capacidade de caraterizar um universo de discurso na área da logística de distribuição de encomendas.

Inicialmente, na secção **Preliminares** são indicados os requisitos iniciais para o desenvolvimento do sistema pretendido.

Posteriormente será descrito o trabalho, desde os predicados de evolução de conhecimento implementados até à base de conhecimento inicial. Serão também apresentadas as **Funcionalidades** implementadas e a sua utilização no contexto do nosso sistema, e todos os restantes requisitos impostos no enunciado.

Por fim, apresentamos conclusões sobre a realização deste trabalho em equipa e sobre a implementação final e resultados obtidos.

3 Preliminares

Na primeira fase de desenvolvimento do sistema é necessário reunir os requisitos fundamentais para proceder à implementação do programa. Uma vez que o sistema tem como objetivo representar a logística de distribuição de entregas, o grupo definiu os principais intervenientes:

```
:- dynamic cliente/2

:- dynamic estafeta/3

:- dynamic encomenda/3

:- dynamic entrega/9

:- dynamic freguesia/2.

:- dynamic veiculo/2.
```

Podemos ainda tirar partido do predicado *solucoes* que é um alias do $findall$:

```
solucoes(T,Q,S) :- findall(T,Q,S).
```

Posto isto, estamos aptos a prosseguir à implementação do programa.

4 Descrição do Trabalho e Implementação

4.1 Caracterização de Conhecimento

- Cliente: $\#IdCliente$, $\#IdFreguesia$ -> V,F
- Estafeta: $\#IdEstafeta$, $Pontuacao$,$Nome$ -> V,F
- Encomenda: $\#IdEncomenda$, $Peso$, $Volume$ -> V,F
- Entrega: $Data$, $Prazo$, $\#IdEntrega$, $\#IdEncomenda$, $\#IdCliente$, $\#IdEstafeta$, $\#IdVeiculo$, $Custo$, $PontuacaoEntrega$ -> V,F
- Veiculo: $\#IdVeiculo$, $TipoVeiculo$ -> V,F
- Freguesia: $\#IdFreguesia$, $NomeFreguesia$ -> V,F

Um cliente possui $IdCliente$ e um identificador da $IdFreguesia$ onde mora e pode realizar encomendas(identificadas por $IdEncomenda$) que possuem um determinado peso e volume.Um determinado $Estafeta$ fará a entrega da encomenda numa determinada $Data$, usando um veículo identificado por $IdVeiculo$. Cada entrega tem um custo associado que depende das características da encomenda (Peso,Volume) bem como do veículo utilizado (Bicleta,Mota,Carro) e do prazo de entrega. O grupo considerou, por exemplo,que menores prazo de entrega significa maior custo de entrega e que diferentes veiculos realizam entregas com diferentes custos. Os critérios de avaliação de custos consideram ainda outros aspetos como Veículo utilizado, e serão explicados nas próximas secções. Por fim, uma entrega tem $PontuacaoEntrega$, que é atribuída pelo cliente.

4.2 Predicados gerais/auxiliares

Nesta secção irão ser expostos todos os predicados auxiliares aos requisitos neste projeto.

- **igual**
  Predicado que por verificar se duas datas são iguais.

```
igual(data(DD,MM,AA),data(DD,MM,AA)).
```

- **anterior**
  Predicado que verifica se uma data é anterior a outra.

```
anterior(data(_,_,A1),data(_,_,A2)) :- A1 < A2.
anterior(data(_,M1,A1),data(_,M2,A2)) :- A1 == A2, M1 < M2.
anterior(data(D1,M1,A1),data(D2,M2,A2)) :- A1 == A2,
            M1 == M2, D1 =< D2.
```

- **posterior**
  Predicado que verifica se uma data é futura a outra.

```
posterior(data(_,_,A1),data(_,_,A2)) :- A1 > A2.
posterior(data(_,M1,A1),data(_,M2,A2)) :- A1 == A2, M1 > M2.
posterior(data(D1,M1,A1),data(D2,M2,A2)) :- A1 == A2,
           M1 == M2, D1 >= D2.
```

- **count**
  Predicado que conta o número de ocorrências de um elemento numa lista.

```
count(_, [], 0).
count(X, [X | T], N) :-!, count(X, T, N1),N is N1 + 1.
count(X, [_ | T], N) :- count(X, T, N).
```

- **sumlist**
  Predicado que faz o somatório dos elementos de uma lista.

```
sum_list([], 0).
sum_list([H|T], Sum) :-
            sum_list(T, Rest),
            Sum is H + Rest.
```

- **pertence**
  Predicado que verifica se um elemento pertence a uma lista.

```
pertence(H,[H|_]):-!,true.
pertence(X,[H|T]) :-
            X \= H,
            pertence(X,T).
```

- **indexOf**
  Predicado responsável por obter o elemento num índice de uma lista.

```
indexOf([Element|_], Element, 0):- !.
indexOf([_|Tail], Element, Index):-
            indexOf(Tail, Element, Index1),
            !,
            Index is Index1+1.
```

- **head**
  Retorna cabeça de uma lista.

```
head([H],H).
head([H|_],H).
```

- **append**
  Predicado responsável por concatenar/unir uma lista.

```
append([ ], L, L).
append([H|L1], L2, [H|L3]):- append(L1, L2, L3).
```

- **comprimento**
  Predicado auxiliar que calcula o comprimento de uma lista.

```
comprimento([],0).
comprimento([_|T],R) :- comprimento(T,N), R is N+1.
```

- **diferentes**
  Predicado que calcula a lista de input sem elementos repetidos.

```
diferentes( [],[] ).
diferentes( [X|L],[X|NL] ) :- removerElemento( L,X,TL ), diferentes( TL,NL ).
```

- **removerElemento**
  Predicado responsável por remover um elemento da lista.

```
removerElemento([ ],_,[ ]).
removerElemento([X|L],X,NL) :- removerElemento( L,X,NL ).
removerElemento([X|L],Y,[X|NL]) :- X \== Y, removerElemento( L,Y,NL ).
```

- **away**
  Predicado responsável por remover o elemento num determinado índice da lista.

```
away([_|H],0,H):-!.
away([G|H],N,[G|L]):- N >= 1, N1 is N - 1,!,away(H,N1,L).
```

- **filtra_lista**
  Predicado auxiliar que recebe uma lista e duas datas e filtra a lista, ficando a conter só as datas que se encontram entre as duas anteriores.

```
filtra_lista([],_,_,[]).
filtra_lista([X|T],D1,D2,[X|T2]) :- entre_datas(D1,D2,X),
                     filtra_lista(T,D1,D2,T2).
filtra_lista([X|T],D1,D2,R) :- not(entre_datas(D1,D2,X)),
    filtra_lista(T,D1,D2,R).
```

4.3 Base de Conhecimento inicial

Para a realização do trabalho, é necessário que o programa já possua uma Base de Conhecimento inicial, para podermos proceder aos testes das funcionalidades do nosso programa. Também estão definidos predicados que permitem fazer uma ampliação da base de conhecimento, nomeadamente o *registaEntrega*, *registaCliente*, *registaEstafeta*, *registaEncomenda* e o *registaFreguesia*.

```
cliente(1, 0).
cliente(2, 1).
cliente(3, 2).
cliente(4, 3).
cliente(5, 4).
cliente(6, 5).
cliente(7, 2).
cliente(8, 1).
cliente(9, 3).
cliente(10, 2).
cliente(11, 4).
cliente(12, 0).
```

```
estafeta(0, 4.1 ,'Daniel').
estafeta(1, 3.9 ,'Nuno').
estafeta(2, 3.5 ,'Guilherme').
estafeta(3, 2.3 ,'Rodrigo').
estafeta(4, 5.0 ,'Joo').
```

```
encomenda(0, 26.5, 10).
encomenda(1, 5.2, 15).
encomenda(2, 22.1, 30).
encomenda(3, 15.3, 12).
encomenda(4, 22, 16).
encomenda(5, 7.4, 30).
encomenda(6, 10, 19).
encomenda(7, 2, 14).
encomenda(8, 25, 33).
encomenda(9, 3, 12).
encomenda(10, 26, 42).
encomenda(11, 5.5,4).
encomenda(12, 7,11).
encomenda(13, 22, 40).
encomenda(14, 1, 2).
encomenda(15, 7, 4.5).
encomenda(16, 25,55).
encomenda(17, 16, 22).
encomenda(18, 1.5, 2.5).
encomenda(19, 18, 14).
encomenda(20, 5.5, 12).
```

```
entrega(data(12,2,2021), 2.0, 1, 20, 2,1,2,11,4).
entrega(data(3,6,2021), 6.5, 2, 19, 3,2,2,7,3).
entrega(data(28,6,2021), 9.0, 3, 18, 1,4,1,5,4).
entrega(data(1,8,2021), 5.0, 4, 17, 6,3,2,7,2).
entrega(data(5,4,2021), 7.5, 5, 16, 8,0,3,9,4).
entrega(data(1,1,2021), 1.5, 6, 15, 9,2,2,11,3).
entrega(data(5,7,2021), 24.0, 7, 14, 10,0,1,5,2).
entrega(data(12,2,2021), 12.0, 8, 13, 11,1,3,8,4).
entrega(data(29,12,2021),4.5, 9, 12, 5,3,2,9,1).
entrega(data(20,10,2021),3.0, 10, 11, 4,4,2,9,5).
entrega(data(4,4,2021), 5.5, 11, 10, 7,1,3,9,3).
entrega(data(1,7,2021), 0.5, 12, 9, 12,3,1,10,4).
entrega(data(4,6,2021), 6.0, 13, 8, 7,3,3,9,3).
entrega(data(19,11,2021),9.0, 14, 7, 3,0,1,5,4).
entrega(data(7,10,2021), 1.5, 15, 6, 12,4,2,9,5).
entrega(data(5,9,2021), 48.0, 16, 5, 4,1,2,6,5).
entrega(data(22,4,2021), 1.0, 17, 4, 1,2,3,13,3).
entrega(data(19,8,2021), 6.5, 18, 3, 7,4,2,7,4).
entrega(data(9,9,2021), 14.0, 19, 2, 4,0,3,8,5).
entrega(data(27,6,2021), 7.0, 20, 1, 6,1,3,9,1).
```

```
freguesia(0,'Braga').
freguesia(1,'Porto').
freguesia(2,'Sao Vicente').
freguesia(3,'Lamacaes').
freguesia(4,'Gualtar').
freguesia(5,'Ferreiros').
```

```
veiculo(1,'Bicicleta').
veiculo(2,'Mota').
veiculo(3,'Carro').
```

4.4 Cálculo dos Custos de Entrega

Nesta secção apresentamos a tabela referente aos custo de uma entrega. O custo de uma entrega varia consoante:

- Veículo utilizado *
- Prazo de entrega **

* A escolha de um veículo é realizada através do peso da encomenda a ser entregue: diferentes veiculos têm custos associados diferentes.
** Prazos de entrega menores equivalem a custos de entrega mais elevados.

Prazo(h) / Veiculo	$<= 2$	$2-5$	$5-8$	$>= 8$
Bicicleta	10	8	6	5
Mota	11	9	7	6
Carro	13	11	9	8

Tabela 1: Tabela de custos das entregas em função do Prazo de Entrega e Veículo

```
preco_veiculo(IdV,R) :- IdV==1, R is 2.
preco_veiculo(IdV,R) :- IdV==2, R is 3.
preco_veiculo(IdV,R) :- IdV==3, R is 5.

preco_prazo(Prazo,R) :- Prazo =<2, R=8.
preco_prazo(Prazo,R) :- Prazo >2, Prazo =< 5, R=6.
preco_prazo(Prazo,R) :- Prazo >5, Prazo =< 8, R=4.
preco_prazo(Prazo,R) :- Prazo >8, R=3.

busca_peso(IdEnc,R) :- solucoes(Peso,(encomenda(IdEnc, Peso,_)),List),
                       head(List,R).

det_veiculo(Peso,R) :- Peso =<5, R is 1.
det_veiculo(Peso,R) :- Peso >5, Peso =< 20, R is 2.
det_veiculo(Peso,R) :- Peso > 20, Peso =< 100, R is 3.
```

Cada veículo tem uma taxa de entrega associada, bem como existem taxas para as diferentes faixas de prazo de entrega. Os custos de entrega são calculados automaticamente por predicados definidos pelo grupo.

Consoante os limites de peso estabelecidos no enunciado do trabalho prático, a escolha de veiculo tem em conta o peso da encomenda em questão, ou seja, o transporte através de bicicleta é privilegiado para encomendas até 5kg, o de mota é privilegiado de 5-20kg e o de carro para encomendas superiores a 20kg e inferiores a 100kg.

5 Funcionalidades do Programa

No enunciado deste trabalho prático são pedidos predicados para as seguintes funcionalidades/queries:

1. identificar o estafeta que utilizou mais vezes um meio de transporte mais ecológico;
2. identificar que estafetas entregaram determinada(s) encomenda(s) a um determinado cliente;
3. identificar os clientes servidos por um determinado estafeta;
4. calcular o valor faturado pela Green Distribution num determinado dia;
5. identificar quais as zonas (e.g., rua ou freguesia) com maior volume de entregas por parte da Green Distribution;
6. calcular a classificação média de satisfação de cliente para um determinado estafeta;
7. identificar o número total de entregas pelos diferentes meios de transporte, num determinado intervalo de tempo;
8. identificar o número total de entregas pelos estafetas, num determinado intervalo de tempo;
9. calcular o número de encomendas entregues e não entregues pela Green Distribution, num determinado período de tempo;
10. calcular o peso total transportado por estafeta num determinado dia.

5.1 Funcionalidade 1: identificar o estafeta que utilizou mais vezes um meio de transporte mais ecológico

A empresa Green Distribution tem como objetivo privilegiar sempre o meio de entrega mais ecológico. A empresa tem ao seu dispor diversos meios de transporte: bicicletas, motos e carros, que podem ser classificados pela vertente ecológica.

1. Calculamos, para cada estafeta, as listas de entregas realizadas com os diferentes meios de transporte.

```
entregas_de_estafeta_bicicleta(IdEst,R) :-
    solucoes(entrega(Data,Prazo,IdEntrega,IdEnc,IdC,IdEst,1,Custo,Pe), (estafeta(IdEst,_,_),entrega(Data,Prazo,IdEntrega,IdEnc,IdC,IdEst,1,Custo,Pe)), R).

entregas_de_estafeta_mota(IdEst,R) :-
    solucoes(entrega(Data,Prazo,IdEntrega,IdEnc,IdC,IdEst,2,Custo,Pe), (estafeta(IdEst,_,_),entrega(Data,Prazo,IdEntrega,IdEnc,IdC,IdEst,2,Custo,Pe)), R).
```

2. Calculamos o grau ecológico de cada estafeta. *

```
calcular_escologia_estafeta(IdEstafeta, R):-
    entregas_de_estafeta_bicicleta(IdEstafeta, X),
    comprimento(X,L1),
    entregas_de_estafeta_mota(IdEstafeta,Y),
    comprimento(Y,L2),
    R is (L1*2 + L2 * 1).


calcula_ecologia_recursive(L):- solucoes([estafeta(IdEst, Pont, N),Y], (estafeta(IdEst, Pont, N), calcular_escologia_estafeta(IdEst,Y)), L).

lista_de_estafetas(L):- solucoes(estafeta(IdEst,Pont,N),(estafeta(IdEst, Pont, N)), L).

lista_de_ecologias(L):- solucoes(Y, (estafeta(IdEst, _, _), calcular_escologia_estafeta(IdEst,Y)), L).
```

3. Finalmente determinamos o estafeta com mais pontos ecológicos e obtemos o resultado pretendido para a **funcionalidade 1**

```
query1(R):- lista_de_estafetas(L1),
            lista_de_ecologias(L2),
            max_list(L2,M),
            indexOf(L2,M,I),
            nth0(I, L1, R).
```

* Uma entrega realizada com mota equivale a 1 ponto ecológico para o estafeta, enquanto uma entrega realizada com bicicleta equivale a 2 pontos. Quanto mais pontos, maior é o grau ecológico do estafeta.

5.2 Funcionalidade 2: identificar que estafetas entregaram determinada(s) encomenda(s) a um determinado cliente

Dado um identificador de cliente $IdCliente$ e uma lista de encomendas, calcular uma lista L de estafetas que fizeram as entregas dessas encomendas. Utilizou-se o predicado *solucoes* para obter a lista de estafetas que fizeram as entregas.

```
query2(IdCliente,[X],L) :- solucoes(estafeta(IdEst,Pont,N),(estafeta(IdEst,Pont,N),encomenda(X,_,_),entrega(_,_,_,X,IdCliente,IdEst,_,_,_)),L).
query2(IdCliente,[X|H],L) :- solucoes(estafeta(IdEst,Pont,N),(estafeta(IdEst,Pont,N),encomenda(X,_,_),entrega(_,_,_,X,IdCliente,IdEst,_,_,_)),L1),
                             query2(IdCliente,H,L2),
                             append(L1,L2,L).
```

5.3 Funcionalidade 3: identificar os clientes servidos por um determinado estafeta

Dado um identificador de estafeta $IdEstafeta$, calcular uma lista L de clientes que foram servidos por esse estafeta. Utilizou-se o predicado *solucoes* para obter a lista de clientes servidos pelo estafeta, seguido do predicado $diferentes$ para filtrar os clientes repetidos.

```
query3(IdEst,L) :- solucoes(cliente(IdC,Freguesia), (estafeta(IdEst,_,_),cliente(IdC,Freguesia),entrega(_,_,_,_,IdC,IdEst,_,_,_)), X),
    diferentes(X,L).
```

5.4 Funcionalidade 4: calcular o valor faturado pela Green Distribution num determinado dia

Dada uma data $data(DD, MM, AA)$, calcular o valor R faturado pela Green Distribution nesse determinado dia.

Estratégia para resolver o problema:

1. Obter lista de Custos das Entregas realizadas no dia em causa.

2. Utilizar a função auxiliar *sum_list* para fazer o somatórios dos custos de entrega.

```
query4(data(DD,MM,AA),R):- solucoes(Custo,(entrega((data(DD,MM,AA)),_,_,_,_,_,_,Custo,_)),L),
                            sum_list(L, R).
```

5.5 Funcionalidade 5: identificar quais as zonas (e.g., rua ou freguesia) com maior volume de entregas por parte da Green Distribution

Obter as N zonas com maior volume de entregas.

Estratégia para resolver o problema:

1. Obter lista de freguesias onde existe pelo menos 1 entrega, através do predicado *solucoes*.
2. Calcular num_entregas em cada freguesia (lista de ocorrências, guarda esse número para cada freguesia).
3. Obter os N maiores num_entregas.
4. As N zonas com maior volume são as freguesias que contém os maiores valores.

```
query5(N,R):-solucoes(freguesia(IdF,Nome),(freguesia(IdF,Nome),entrega(_, _, _,_, IdCliente, _, _, _,_),cliente(IdCliente, IdF)),LFreg1),
            diferentes( LFreg1,LFreg2),
            lista_ocorrencias(LFreg1,LFreg2,Ocorrencias),
            busca(N,Ocorrencias,LFreg2,R).

busca(0,_,_,[]).
busca(N,Ocorrencias,Lista,[X|T]):- comprimento(Lista,C), N =< C ,max_list(Ocorrencias,M),
                            nth0(I,Ocorrencias,M),
                            nth0(I,Lista,X),
                            away(Ocorrencias,I,Ocorrencias2),
                            away(Lista,I,Lista2),
                            N1 is N-1,
                            busca(N1,Ocorrencias2,Lista2,T).
```

5.6 Funcionalidade 6: calcular a classificação média de satisfação de cliente para um determinado estafeta

Estratégia para resolver o problema:

1. Através do predicado *solucoes*, obter lista de Pontuacoes atribuídas ao estafeta nas suas entregas.
2. Calcular Nº Entregas realizadas pelo estafeta.
3. Fazer o somatório das pontuações das entregas.
4. Finalmente, calcular a classificação média de satisfação de cliente para o estafeta.

```
lista_de_pontuacoes_estafeta(IdEst,L) :- solucoes(Pe, (estafeta(IdEst,_,_), entrega(_,_,_,_,_,IdEst,_,_,Pe)), L).

query6(IdEst,R):- lista_de_pontuacoes_estafeta(IdEst,X),
    comprimento(X,L),
    sum_list(X,S),
    R is S / L.
```

5.7 Funcionalidade 7: identificar o número total de entregas pelos diferentes meios de transporte, num determinado intervalo de tempo

Dadas duas datas que definem um intervalo de tempo, obter o número total de entregas pelos diferentes meios de transporte nesse intervalo de tempo.

Estratégia para resolver o problema:

1. Através do predicado *solucoes*, obter 3 listas de Datas de entrega uma para veículo(Bicicleta, Mota e Carro).
2. Filtrar essas listas, obtendo apenas entregas que se emcontram dentro do intervalo de tempo pretendido.
3. Através do predicado *comprimento*, obter o número de entregas realizadas de bicicleta, mota e carro no intervalo de tempo pretendido.

```
query7(data(DD1,MM1,AA1),data(DD2,MM2,AA2),Bicicleta,Mota,Carro) :-solucoes(D,(entrega(D,_,_,_,_,_,1,_,_)),EBicicleta),
                                                                   solucoes(D,(entrega(D,_,_,_,_,_,2,_,_)),EMota),
                                                                   solucoes(D,(entrega(D,_,_,_,_,_,3,_,_)),ECarro),
                                                                   filtra_lista(EBicicleta,data(DD1,MM1,AA1),data(DD2,MM2,AA2),X1),
                                                                   filtra_lista(EMota,data(DD1,MM1,AA1),data(DD2,MM2,AA2),X2),
                                                                   filtra_lista(ECarro,data(DD1,MM1,AA1),data(DD2,MM2,AA2),X3),
                                                                   comprimento(X1,Bicicleta),
                                                                   comprimento(X2,Mota),
                                                                   comprimento(X3,Carro).
```

5.8 Funcionalidade 8: identificar o número total de entregas pelos estafetas, num determinado intervalo de tempo

Estratégia para resolver o problema:

1. Através do predicado *solucoes*, obter lista de Datas de todas as entregas.
2. Filtrar essa lista, eliminando as datas das entregas que não se encontram dentro do intervalo de tempo pretendido.
3. Através do predicado *comprimento*, obter o número total de entregas nesse intervalo de tempo.

```
query8(data(DD1,MM1,AA1),data(DD2,MM2,AA2),R) :-solucoes(D,(entrega(D,_,_,_,_,_,_,_,_)),X1),
                                                filtra_lista(X1,data(DD1,MM1,AA1),data(DD2,MM2,AA2),X2),
                                                comprimento(X2,R).

entre_datas(D1,D2,D3):-anterior(D1,D3),posterior(D2,D3).
```

5.9 Funcionalidade 9: calcular o número de encomendas entregues e não entregues pela Green Distribution, num determinado período de tempo

Estratégia para resolver o problema:

1. Através do predicado *solucoes* e do *comprimento*, obter o número total de entregas.
2. Filtrar com o predicado *filtra_lista* a lista que contém todas as entregas,calcular o comprimento da lista filtrada e obter o número de entregas entregues.
3. Finalmente, para calcular o número de não entregues subtrair ao total de entregas as que foram entregues.

```
query9(data(DD1,MM1,AA1),data(DD2,MM2,AA2),NE,E):-solucoes(D,(entrega(D,_,_,_,_,_,_,_,_)),All),
                                                   comprimento(All,A),
                                                   solucoes(Data,(entrega(Data,_,_,IdEnc,_,_,_,_,_),encomenda(IdEnc,_,_)),Entregues),
                                                   filtra_lista(Entregues,data(DD1,MM1,AA1),data(DD2,MM2,AA2),X2),
                                                   comprimento(X2,E),
                                                   NE is A-E.
```

5.10 Funcionalidade 10: calcular o peso total transportado por estafeta num determinado dia

Estratégia para resolver o problema:

1. Através do predicado *solucoes*, obter lista de Pesos das entregas realizadas pelo estafeta no dia pretendido.
2. Obter o total de peso transportado através do predicado *sum_list*.

```
query10(IdEst,data(DD,MM,AA),R) :- solucoes(P,(entrega(data(DD,MM,AA),_,_,IdEnc,_,IdEst,_,_,_),estafeta(IdEst,_,_),encomenda(IdEnc,P,_)),X1),
                                   sum_list(X1,R).
```

5.11 Exemplos de Resultados das Funcionalidades

Seguem-se exemplos de invocações das funcionalidades e os respetivos outputs:

```
?- query1(R).
R = estafeta(4, 5.0, 'João').

?- query2(7,[10,8,3],R).
R = [estafeta(1, 3.9, 'Nuno'), estafeta(3, 2.3, 'Rodrigo'), estafeta(4, 5.0, 'João')] .

?- query3(1,R).
R = [cliente(2, 1), cliente(4, 3), cliente(6, 5), cliente(7, 2), cliente(11, 4)].

?- query4(data(12,2,2021),R).
R = 19.

?- query5(3,R).
R = [freguesia(2, 'São Vicente'), freguesia(0, 'Braga'), freguesia(3, 'Lamaçães')] .

?- query6(2,R).
R = 3.

?- query7(data(1,2,2021),data(31,5,2021),Bicicleta,Mota,Carro).
Bicicleta = 0,
Mota = 1,
Carro = 4 .

?- query8(data(1,1,2021),data(31,5,2021),R).
R = 6 .

?- query9(data(1,1,2021),data(31,5,2021),NE,E).   NE = 14,
E = 6 .

?- query10(1,data(12,2,2021),R).
R = 27.5.
```

Obtivemos os resultados esperados para todas as funcionalidades implementadas.

6 Conclusão

Tendo este projeto como base a linguagem de programação *PROLOG*, e sendo esta nova para todos os elementos do grupo, este trabalho prático permitiu aprofundar e consolidar todo o conhecimento obtido durante as aulas práticas da unidade curicular de Inteligência Artificial.

Durante a execução e implementação do projeto, a maior dificudade encontrada pelo grupo foi definir a estratégia para implementar os diversos objetivos propostos no enunciado.

Por fim, obtivemos os resultados esperados e pensamos ter respondido corretamente aos problemas apresentados, não só concluindo-os como também melhorando o método de resolução desses problemas ao longo das sessões.