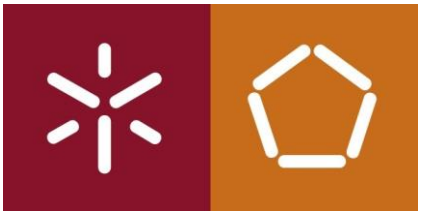

Universidade do Minho
Escola de Engenharia
Licenciatura em Engenharia informática

Unidade Curricular de Inteligência Artificial

Ano Letivo de 2021/2022

Trabalho Prático de Inteligência Artificial

Grupo 19

José Neiva, 92945

João Martins, 91669

Jorge Lima, 93183

Rúben Santos, 93257

Índice

Conteúdo

1. Introdução

Este trabalho prático foi resolvido com recurso ao conhecimento adquirido nas aulas da UC e material bibliográfico digital sobre Prolog. O objetivo deste projeto é a utilização desta linguagem no âmbito da representação de conhecimento e construção de mecanismos de raciocínio para a resolução de problemas.

Prolog é uma linguagem declarativa de lógica de primeira-ordem, com uma sintaxe base semelhante a outras linguagens de programação populares, mas tem algumas diferenças chave que a diferenciam.

2. Base de Conhecimento

A base de conhecimento contém os factos necessários para o desenvolvimento do nosso programa. Foi gerada e posteriormente alterada tendo em conta o conhecimento que era necessário guardar, dependendo dos nossos objetivos.

2.1. Estafeta

Um estafeta é caracterizado pelo seu Nome, Avaliação Total, Número de encomendas, e o Estado (ocupado ou não), a Avaliação Total é a soma de todas as avaliações feitas ao estafeta, este valor a dividir pelo número de encomendas dará a avaliação relativa que será um fator para a quantidade de encomendas que determinado estafeta recebe, quanto maior este valor mais encomendas lhe é pedido.

```
%-------------------------estafetas---------------------------------
%estafeta(nome,avaliacao total, numero de ecomendas,ocupado)
estafeta(joao,0,0,true).
estafeta(jorge,0,0,false).
estafeta(ruben,0,0,false).
estafeta(neiva,0,0,false).
```

2.2. Transporte

O transporte é caracterizado pelo tipo de meio de transporte, Peso, Velocidade media, Índice de poluição e ocupado, no entanto, de modo a não duplicar dados criamos um facto à parte que é denominado por specs_transporte, este contem todas as características técnicas de cada tipo. Assim cada meio de transporte do nosso sistema apenas precisa de ter a ele associado um booleano, para saber se está ocupado. Foi considerado a adição de um id a cada um, mas de momento o nosso sistema não necessita de diferenciar dois meios de transporte do mesmo tipo.

```
%-------------------------transporte---------------------------------
%specs_transporte(tipo,peso,velocidade media,índice de poluição)
specs_transporte(bicicleta,5,10,1).
specs_transporte(moto,20,35,2).
specs_transporte(carro,100,3,3).

%transporte(tipo,ocupado)
transporte(bicicleta,true).
transporte(bicicleta,false).
transporte(moto,false).
transporte(moto,false).
transporte(carro,false).
transporte(carro,false).
```

2.3. Mapa

O mapa é um grafo, no qual, cada aresta contém uma origem, um destino e a distância em quilómetros entre eles. O início corresponde à localização da empresa Green Distribution. Criamos um grafo baseado em dados imaginários, para exemplificar uma possível utilização deste sistema.

```
%-------------------------mapa---------------------------------------
%mapa(origem,destino,distancia(km))
inicio(santa_marta_de_portuzelo).
mapa(cardielos,nogueira,5).
mapa(sarreleis,cardielos,2.5).
mapa(santa_marta_de_portuzelo,sarreleis,2.5).
mapa(santa_marta_de_portuzelo,perre,5).
mapa(perre,outeiro,5).
mapa(santa_marta_de_portuzelo,meadela,5).
mapa(meadela,darque,10).
mapa(darque,vila_de_punhe,10).
```

2.4. Outros factos

O primeiro facto representa o peso de cada fator no cálculo do preço de uma encomenda, por exemplo por cada quilometro o cliente paga 0.5€.

O facto n_encomendas tem o número total de encomendas e também representa o id da próxima encomenda. No início do sistema temos zero encomendas, mas vão ser adicionadas de acordo com a estrutura do comentário na imagem abaixo, que contém toda a informação de uma encomenda.

```
%peso de cada fator na formula que calcula o peso
%preco(distancia(km),peso(Kg),volume(m³),1/prazo,bicicleta,moto,carro)
preco(0.5,0.5,0.5,1,1,2,3).

n_encomendas(0).
%encomenda(cliente,id,peso(kg),volume(m^3),prazo(Horas),preco,freguesia,data(Timestamp s),
%estafeta,transporte,(estado-boolean entregue,a entregar)).
```

3. Estrutura Geral

O modulo Funcionalidades é o mais importante do programa, é onde existe o "coração" do programa, onde tem todas os predicados das principais funcionalidades do programa.

3.1. Predicado Update

Dois predicados muito similares, responsáveis por atualizar estados. O True atualiza o estado quando o programa recebe uma nova encomenda. O False atualiza o estado quando o utilizador da entrega de uma encomenda.

```
updateallTrue(Transporte,Estafeta,Id) :-
    remove(Transporte),remove(Estafeta),remove(Id),
    addNewTrue(Transporte,Estafeta,Id).

updateallFalse(Transporte,Estafeta,Encomenda,Avaliacao) :-
    remove(Transporte),remove(Estafeta),remove(Encomenda),
    addNewFalse(Transporte,Estafeta,Encomenda,Avaliacao).
```

3.2. Predicado Caminho e isZona

O predicado caminho simplesmente vai encontrar um caminho do ponto A ao ponto B, isto se for possível.

O predicado isZona verifica se um local existe, na nossa base de conhecimento.

```
%Calcula se existe caminho do ponto A ate ao B
caminho(A,B,P,Km) :- caminho1(A,[B],P,Km).
caminho1(A,[A|P1],[A|P1],0).
caminho1(A,[Y|P1],P,K1) :-
  adjacente(X,Y,Ki),
  \+(member(X,[Y|P1])),
  caminho1(A,[X,Y|P1],P,K), K1 is K + Ki.

%para verificar se o local existe
isZona(A) :- mapa(A,_,_).
isZona(A) :- mapa(_,A,_).
```

3.3. Predicados escolhe transporte/estafeta

O predicado escolhe transporte verifica qual é o melhor tipo de transporte para uma encomenda com determinadas características. Começa por verificar todos os transportes não ocupados e ordena por ordem crescente de índice de poluição.

Posteriormente, verifica se o transporte tem peso máximo e velocidade média suficiente para deslocar uma determinada encomenda dentro do prazo. Se não conseguir testa o próximo transporte com o menor índice de poluição.

O predicado que escolhe o estafeta, verifica os estafetas não ocupados coloca-os numa lista ordenada pela avaliação de cada um e vai buscar a cabeça dessa mesma lista.

```
escolhetransporte(Peso,Distancia,Prazo,R) :-
  findall((X,Indice),(transporte(X,false),specs_transporte(X,_,_,Indice)),Y),sort(2,@=<,Y,Transportes),
  escolhetransporte_aux(R,Transportes,Peso,Distancia,Prazo).

escolhetransporte_aux(H,[(H,_)],Peso,Distancia,Prazo) :-
    specs_transporte(H,P,Velocidade,_),
    P >= Peso, Velocidade >= (Distancia/Prazo),!.
escolhetransporte_aux(Nome,[(Nome,_)|_],Peso,Distancia,Prazo) :-
    specs_transporte(Nome,P,Velocidade,_),
    P >= Peso, Velocidade >= (Distancia/Prazo),!.
escolhetransporte_aux(Nome,[_|T],Peso,Distancia,Prazo) :- escolhetransporte_aux(Nome,T,Peso,Distancia,Prazo).
```

3.4. Handlers

O Handler do menu entrega encomenda, recebe os dados fornecidos pelo estafeta que entregou a encomenda, valida a data fornecida, faz a conversão do prazo para segundos, verifica se há atraso na entrega e atualiza os estados do transporte, do estafeta e da encomenda.

```
entregaEncomendaHandler(Id,Ano,Mes,Dia,Hora,Minutos,Avaliacao):-
    encomenda(_,Id,_,_,Prazo,_,_,Data,Estafeta,Transporte,False),
    validadata(date(Ano,Mes,Dia,Hora,Minutos,0,0,-,-)),
    date_time_stamp(date(Ano,Mes,Dia,Hora,Minutos,0,0,-,-), TimeStamp), PrazoS is Prazo*3600,Data < TimeStamp,
    (Data+PrazoS) >= TimeStamp,
    updateallFalse(transporte(Transporte,true),estafeta(Estafeta,_,_,true)
        ,encomenda(_,Id,_,_,Prazo,_,_,Data,Estafeta,Transporte,False),Avaliacao),
    write('A encomenda foi entregue sem atrasos, a avalicao foi: '),writeln(Avaliacao).

entregaEncomendaHandler(Id,Ano,Mes,Dia,Hora,Minutos,Avaliacao):-
    encomenda(_,Id,_,_,Prazo,_,_,Data,Estafeta,Transporte,False),
    validadata(date(Ano,Mes,Dia,Hora,Minutos,0,0,-,-)),
    date_time_stamp(date(Ano,Mes,Dia,Hora,Minutos,0,0,-,-), TimeStamp), PrazoS is Prazo*3600,Data < TimeStamp,
    (Data+PrazoS) < TimeStamp,divisao(Avaliacao,2,NewAV),
    updateallFalse(transporte(Transporte,true),estafeta(Estafeta,_,_,true)
        ,encomenda(_,Id,_,_,Prazo,_,_,Data,Estafeta,Transporte,False),NewAV),
    write('A encomenda foi entregue com atrasos, a avalicao leva penalizacao de 50%, avalicao é: '),writeln(NewAV).

entregaEncomendaHandler(Id,Ano,Mes,Dia,Hora,Minutos,_) :-
    encomenda(_,Id,_,_,_,_,_,Data,_,_,False),
    validadata(date(Ano,Mes,Dia,Hora,Minutos,0,0,-,-)),
    date_time_stamp(date(Ano,Mes,Dia,Hora,Minutos,0,0,-,-), TimeStamp),
    Data > TimeStamp, writeln('A data inserida nao e uma data valida.').
```

O Handler que trata da criação da encomenda, começa por obter a data atual através do predicado get_time(), verifica se o caminho é possível calculando, também, a distancia, escolhe o transporte mais adequado às características da encomenda, escolhe os estafeta, calcula o preço da encomenda e insere a encomenda na base de dados por fim, da update da base de conhecimento.

Caso por uma ou mais razões não seja possível efetuar a entrega informamos o utilizador.

```
fazEncomendaHandler(Nome,Peso,Volume,Prazo,Freguesia) :-
    get_time(TimeStamp),
    caminho(santa_marta_de_portuzelo,Freguesia,_,Distancia),
    escolhetransporte(Peso,Distancia,Prazo,Transporte),
    escolheestafeta(Estafeta),
    n_encomendas(Id),
    calculapreco(Distancia,Peso,Volume,Prazo,Transporte,Preco),
    insere(encomenda(Nome,Id,Peso,Volume,Prazo,Preco,Freguesia,TimeStamp,Estafeta,Transporte,false)),
    updateallTrue(transporte(Transporte,false),estafeta(Estafeta,_,_,false),n_encomendas(Id)),!,
    write('O id da sua encomenda é: '),writeln(Id),
    write('A sua encomenda sera entregue por: '),writeln(Estafeta),
    write('Modo de Transporte: '),writeln(Transporte),
    write('Preco Total: '),writeln(Preco).

fazEncomendaHandler(_,_,_,_,_) :- writeln('Pedimos desculpa mas não é possivel fazer a sua encomenda.').
```

4. Funcionalidades

4.1. Estafeta mais ecológico.

Funcionalidade: Identificar o estafeta que utilizou mais vezes um meio de transporte mais ecológico.

Para encontrar o estafeta mais ecológico vamos usar os índices de poluição de cada meio de transporte. Primeiro vamos encontrar todos os estafetas que fizeram pelo menos uma encomenda, e de seguida procuramos as médias de índice de poluição por encomenda de cada um, usando uma regra auxiliar.

Esta regra (quantas_vezes_usou), permite para um dado estafeta calcular o seu índice de poluição médio por encomenda.

Sendo assim apenas necessitamos de escolher a menor destas médias e associado a esta temos o estafeta mais ecológico.

```
estafeta_mais_ecologico(Mais_ecologico) :-
    findall(Estafeta,(estafeta(Estafeta,_,Num_encomendas,_) , Num_encomendas =\= 0),L_estafeta),
    maplist(quantas_vezes_usou,L_estafeta,L_pares),
    min_member((_,Mais_ecologico),L_pares).

/* função auxiliar a querie 1, que dado um nome de um estafeta
   calcula o seu indice de poluição por encomenda,
   gera o par (indice de poluição por encomenda,estafeta).*/
quantas_vezes_usou(Nome_estafeta,(Pol_media,Nome_estafeta)) :-
    aggregate_all((sum(Ind_poluicao),count),
        (specs_transporte(Meio_transporte,_,_,Ind_poluicao)
            ,encomenda(_,_,_,_,_,_,_,_,Nome_estafeta,Meio_transporte,true)),
        (Ind_poluicao_total,Num_encomendas)),
        Pol_media is Ind_poluicao_total/Num_encomendas.
```

4.2. Estafetas que atenderam um cliente.

Funcionalidade: Identificar que estafetas entregaram determinada(s) encomenda(s) a um determinado cliente.

Com este objetivo em mente, começamos por encontrar todos os estafetas que atenderam este cliente, incluindo todos numa lista e retirando os duplicados.

A partir daí apenas temos que encontrar que encomendas foram resolvidas para cada estafeta. Assim criamos uma regra auxiliar, que associa um Estafeta aos Id's das encomendas ligadas a este e ao cliente pedido. Aplicando esta regra a cada um dos estafetas da lista acima referida, obtemos o resultado pretendido.

```
estafetas_que_servem(Cliente,Lista) :-
    findall(Estafeta,encomenda(Cliente,_,_,_,_,_,_,_,Estafeta,_,_),Lista_dup),
    sort(Lista_dup,Lista_Est),
    maplist(que_encomendas(Cliente),Lista_Est,Lista).

/* função auxiliar a querie 2, que dado um nome de um estafeta e de um cliente
   encontra os ids dos pedidos associados aos dois,
   gera o par (Estafeta,[id]).*/
que_encomendas(Cliente,Estafeta,(Estafeta,Lista_Ids)) :-
    findall(Ids,encomenda(Cliente,Ids,_,_,_,_,_,_,Estafeta,_,_),Lista_Ids).
```

4.3. Clientes servidos por um estafeta.

Funcionalidade: Identificar os clientes servidos por um determinado estafeta

Esta é bastante simples apenas colocamos uma lista todos os clientes atendidos por este estafeta e depois removemos aqueles que estão repetidos.

```
clientes_servidos(Estafeta,Lista) :-
    findall(Cliente,encomenda(Cliente,_,_,_,_,_,_,_,Estafeta,_,_),Lista_wdup),
    sort(Lista_wdup,Lista).
```

4.4. O valor faturado num determinado dia.

Funcionalidade: Calcular o valor faturado pela Green Distribution num determinado dia.

Devido a maneira como guardamos as datas das encomendas, quando é inserido o dia no nosso programa, convertemos este para um intervalo de timestamps. Sendo assim apenas temos que somar todos os preços das encomendas cuja data se encontra entre o intervalo procurado.

```
numero_total_faturado(Time_stamp_inicial,Time_stamp_final,Valor_faturado) :-
    aggregate_all(sum(Preco),
        (encomenda(_,_,_,_,_,Preco,_,Time,_,_,true),
            mybetween(Time_stamp_inicial,Time_stamp_final,Time)),
        Valor_faturado).
```

4.5. Identificar quais as zonas com maior volume de entregas.

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

Nesta funcionalidade vamos permitir escolher o número de zonas com mais volume que o utilizador deseja visualizar. Para isso, primeiro vamos encontrar todas as zonas diferentes do nosso sistema.

Posteriormente usamos uma regra auxiliar, que associa uma zona ao número de encomendas lá feitas, e aplicamos a todas a zonas. Assim podemos ordenar por ordem decrescente e temos uma lista que desejamos.

Para concluir vamos apenas guardar as melhores n zonas, sendo n o numero de zonas que o utilizador escolheu visualizar.

```
zonas_com_mais_volume(N,Top_n_zones_pair) :-
    findall(Zona,isZona(Zona),Lista_zona_dup),
    sort(Lista_zona_dup,Lista_zona),
    maplist(volume_zona,Lista_zona,Lista_zona_volume),
    sort(1,@>=,Lista_zona_volume,Lista_zona_volume_sorted),
    take(N,Lista_zona_volume_sorted,Top_n_zones_pair).

/* função auxiliar a pergunta 5, que dado um nome de uma zona calcula o numero de vezes que foi usada,
   gera o par (numero de vezes que foi usada,zona).*/
volume_zona(Zona,(Volume,Zona)) :-
    aggregate_all(count,encomenda(_,_,_,_,_,_,Zona,_,_,_,_),Volume).
```

4.6. Classificação média de satisfação de um estafeta.

Funcionalidade: Calcular a classificação média de satisfação de cliente para um determinado estafeta.

Devido a maneira como guardamos a informação na base de conhecimento, esta funcionalidade apenas calcula a divisão entre o número total das avaliações a dividir pelo número de encomendas. Temos apenas que prevenir que não aconteça a divisão por 0.

```
mediaestafeta(Estafeta,Media) :-
    estafeta(Estafeta,Total,Numero,_),
    Numero =\= 0,
    Media is Total/ Numero.
```

4.7. Total de entregas pelos diferentes meios de transporte.

Funcionalidade: Identificar o número total de entregas pelos diferentes meios de transporte, num determinado intervalo de tempo.

Começamos por fazer uma lista sem repetidos com todos os meios de transporte. Depois para cada transporte calculamos em quantas encomendas foram utilizados dentro dos dois timestamps que representam o intervalo de tempo, através de uma regra auxiliar.

```
entregas_meio_transporte(Time_stamp_inicial,Time_stamp_final,Lista) :-
    findall(Meio_transporte,transporte(Meio_transporte,_),Lista_mt_dup),
    sort(Lista_mt_dup,Lista_mt),
    maplist(quantas_vezes_usou_transporte(Time_stamp_inicial,Time_stamp_final),Lista_mt,Lista).

/* função auxiliar a querie 7, que dado um nome de um intervalo de tempo e um meio de trasnporte,
   conta as vezes que foi utilizado nesse intervalo,
   gera o par (Vezes usado,Meio transporte).*/
quantas_vezes_usou_transporte(Time_stamp_inicial,Time_stamp_final,Meio_transporte,
    (Vezes_usado,Meio_transporte)) :-
    aggregate_all(count,
        (encomenda(_,_,_,_,_,_,_,_,Time,_,Meio_transporte,true),
            mybetween(Time_stamp_inicial,Time_stamp_final,Time)),
        Vezes_usado).
```

4.8. Número total de entregas pelos estafetas.

Funcionalidade: Identificar o número total de entregas pelos estafetas, num determinado intervalo de tempo.

Mais uma vez a utilização de timestamps para representar datas torna esta funcionalidade bastante simples, bastando contar o número total de entregas naquele intervalo de tempo.

```
numero_total_entregas(Time_stamp_inicial,Time_stamp_final,Numero_total_entregas) :-
    aggregate_all(count,
        (encomenda(_,_,_,_,_,_,_,_,Time,_,_,true),
            mybetween(Time_stamp_inicial,Time_stamp_final,Time)),
        Numero_total_entregas).
```

4.9. Número de encomendas entregues e não entregues.

Funcionalidade: Calcular o número de encomendas entregues e não entregues pela Green. Distribution, num determinado período de tempo.

Devido a nossa base de conhecimento precisamos de contar ambos um de cada vez, primeiro calculamos as encomendas que ainda não acabaram e depois aquelas que já acabaram.

```
entregas_comp(Time_stamp_inicial,Time_stamp_final,Comp,Ncomp) :-
    aggregate_all(count,
        (encomenda(_,_,_,_,_,_,_,_,Time,_,_,false),
            mybetween(Time_stamp_inicial,Time_stamp_final,Time)),
        Ncomp),
    aggregate_all(count,
        (encomenda(_,_,_,_,_,_,_,_,Time,_,_,true),
            mybetween(Time_stamp_inicial,Time_stamp_final,Time)),
        Comp).
```

4.10. Peso total transportado num determinado dia.

Funcionalidade: Calcular o peso total transportado por estafeta num determinado dia.

Mais uma vez a utilização de timestamps para representar datas torna esta funcionalidade bastante simples, bastando somar o peso total das encomendas naquele intervalo de tempo.

```
peso_total_entrege(Time_stamp_inicial,Time_stamp_final,Peso_total) :-
    aggregate_all(sum(Peso),
        (encomenda(_,_,Peso,_,_,_,_,Time,_,_,true),mybetween(Time_stamp_inicial,Time_stamp_final,Time)),
        Peso_total).
```

5. Menu

Desenvolvemos um menu para facilitar o uso no nosso programa, este é constituído por 9 elementos que o utilizador pode escolher:

1,2,3,7 -> Estas três opções têm funcionalidades muito similares, permitindo ao utilizador consultar todos os estafetas, encomendas ou transporte na nossa base de conhecimento.
4 -> Esta opção permite ao utilizador criar uma encomenda, o utilizador ira indicar o peso, volume, prazo máximo para a entregar se concluída e a freguesia onde a encomenda deve ser entregue.
5 -> Esta opção permitira ao utilizador verificar várias estatísticas como pedido no enunciado.
6 -> Esta opção permite ao utilizador dar como entregue uma encomenda existente na base de conhecimento, o utilizador será interrogado pelo a data de entrega e a avaliação dada pelo cliente.
8 -> Permite ao utilizador modificar a tabela de preço.

```
menu :-
    writeln('--------------------------------MENU-----------------------------------'),nl,
    writeln('1 - consultar os estafetas'),
    writeln('2 - consultar encomendas'),
    writeln('3 - consultar transporte'),
    writeln('4 - adicionar encomenda'),
    writeln('5 - verificar estatisticas encomendas'),
    writeln('6 - Dar entrega da encomenda'),
    writeln('7 - Verificar tabela de precos'),
    writeln('8 - Mudar tabela de precos'),
    writeln('9 - Exit'),
    write('Choose: ').
```

6. Conclusão

Tentamos fazer o projeto sabendo que este é apenas uma primeira fase, sendo assim percebemos que podemos ou temos que mudar o funcionamento do sistema para acomodar novos objetivos propostos para a segunda fase, por isso tentamos generalizar e permitir facilidade de alteração no sistema.