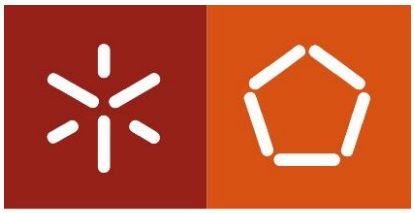

Universidade do Minho

Mestrado Integrado em Engenharia Informática

Inteligência Artificial - Trabalho Prático (Fase 1) - Ano Letivo 2021/2022

Grupo 42

Gonçalo Braz (a93270) Simão Cunha (a93262)
Tiago Silva (a93277) Gonçalo Pereira (a93168)

2 de dezembro de 2021

Conteúdo

1 Introdução

No âmbito da Unidade Curricular de Inteligência Artificial, foi proposto um exercício de forma a que utilizemos a linguagem de programação em lógica PROLOG, no âmbito da representação de conhecimento e construção de mecanismos de raciocínio para a resolução de problemas.

O nosso projeto consiste no desenvolvimento de um sistema de representação de conhecimento e raciocínio com capacidade para caracterizar um universo de discurso na área da logística de distribuição de encomendas. Para tal, elaboramos a nossa base de conhecimento, implementamos funcionalidades de inserção e de remoção de estafetas, meios de transporte, encomendas/pedidos e de clientes e listagens dos mesmos de acordo com um determinado argumento e algumas *queries* pedidas pela equipa docente. Para a 1ª fase do projeto, decidimos criar um menu que possibilitasse ao utilizador uma análise mais agradável das queries e das listagens de termos. Todos estes itens serão explicados nas próximas secções.

2 Implementação da solução

2.1 Base de conhecimento

Começando pelo estudo das entidades que o projeto pretende manipular, o nosso grupo concluiu que necessitaríamos de 3 factos, os **meios de transporte**, o **cliente** e o **pedido**, que estarão todos incluídos na entidade principal **estafeta**.

Vejamos o que constitui cada facto.

2.1.1 Cliente

Um cliente é constituído por:

- `Nome`: nome do cliente;
- `ID`: número único, identificador de um cliente.

2.1.2 Meio de transporte

Um meio de transporte é constituído por:

- `Matrícula`: número único, identificador do meio de transporte;
- `Tipo`: identifica o tipo de transporte, por exemplo, moto;
- `Velocidade`: representa a velocidade máxima a que o veículo pode andar;
- `Peso`: peso máximo de uma encomenda que pode ser transportada por este veículo.

2.1.3 Pedido

Um pedido é constituído por:

- `Cliente`;
- `ID`: número único, identificador do pedido;
- Prazo de entrega: data máxima de entrega do pedido, limitada pelo cliente;
- `Rua`: localização dentro de uma zona, onde deverá ser efetuada a entrega;
- `Zona`: localização, mais abrangente, onde deverá ser efetuada a entrega;
- `Peso`: peso da encomenda;
- `Data de pedido`: data em que o pedido foi realizado;
- `Estado`: indica conclusão da entrega.

2.1.4 Estafeta

Um estafeta é constituído por:

- `Nome`: nome do estafeta;
- `ID`: número único, identificador do estafeta;
- `Freguesia`: zona de entrega atribuída ao estafeta;
- `Meio de transporte`: transporte utilizado pelo estafeta;
- `Classificação`: classificação do estafeta que se divide em dois parâmetros somatório das classificações e número total de classificações;
- `Lista de entregas`: lista de entregas a serem realizadas pelo estafeta;
- `Penalização`: penalização caso o estafeta não cumpra os prazos de entrega determinados.

2.2 Implementação das listagens

Nesta secção, iremos mostrar o nosso raciocínio para listar no ecrã estafetas, meios de transporte e pedidos. Convém referir que a implementação destes predicados é muito semelhante, uma vez que agrupamos os termos numa lista.

2.2.1 Estafetas

O procedimento para a obtenção de estafetas é o seguinte:

1. Utilização do predicado `findall` para procurar todos os estafetas através de um dado argumento e colocá-los numa lista.

   Por exemplo, para listar estafetas com um dado nome, efetuamos da seguinte forma:

```
estafeta_nome(Nome,R) :- findall(estafeta(Nome,ID,Z,MT,Cl,LE,P),
                                 estafeta(Nome,ID,Z,MT,Cl,LE,P),
                                 R).
```

2.2.2 Meios de transporte

O procedimento para a obtenção de meios de transporte é o seguinte:

1. Utilização do predicado `findall` para procurar todos os meios de transporte dos estafetas através de um dado argumento e colocá-los numa lista.

   Por exemplo, para listar meios de transporte com um dado tipo, efetuamos da seguinte forma:

```
meioTransporte_tipo(Tipo,R) :- findall(meio_transporte(M,Tipo,V,P),
                               estafeta(_,_,_,meio_transporte(M,Tipo,V,P),_,_,_),
                               R).
```

2.2.3 Pedidos

O procedimento para a obtenção dos pedidos é o seguinte:

1. Utilização do predicado `findall` para procurar todos os pedidos dos estafetas e colocá-los numa lista.
2. Utilização de um predicado auxiliar que percorre uma lista de listas oriundas do passo 1. para filtrar os pedidos de acordo com um dado argumento.

   Por exemplo, para listar pedidos de um dado cliente, efetuamos da seguinte forma:

```
pedido_cliente(ID_Cli,R) :- findall(LE,
                                    estafeta(_,_,_,_,_,LE,_), LAux),
                            filter_by_IDC(ID_Cli,LAux,[],R).

filter_by_IDC(_,[],R,R).
filter_by_IDC(IDC,[[]|TS], Acc,R) :- filter_by_IDC(IDC,TS, Acc,R).
filter_by_IDC(IDC,[[pedido(cliente(NC,ID),ID_Ped, DE, R, Z, Pes, DP, Est)|T]|TS],Acc,R) :-
(
IDC == ID->
filter_by_IDC(IDC,[T|TS],[pedido(cliente(NC,ID),ID_Ped, DE, R, Z, Pes, DP,Est)|Acc],R);
filter_by_IDC(IDC,[T|TS],Acc,R)
).
```

NOTA: Temos a noção que estes predicados não são muito eficientes para uma extensa base de conhecimento, uma vez que deveríamos adicionar apenas os pedidos certos à medida que vamos executando o `findall`.

2.3 Implementação das queries

2.3.1 Query 1: Identificar o estafeta que utilizou mais vezes um meio de transporte mais ecológico

Para esta primeira pergunta, começamos por determinar uma lista de todos os estafetas registados, esta lista priorizará guardar apenas os estafetas que usem bicicleta mas, em caso de isto não ser possível, tentará fazer o mesmo para os estafetas que usem moto e por último os que usem carro. Seguimos esta ordem pois assim definimos que esta é, de forma descendente, a lista do veículo mais ecológico ao menos.

Determinada esta lista, procuramos encontrar o elemento nela que realizou mais pedidos, sendo este considerado o estafeta que utilizou mais vezes um meio de transporte ecológico.

2.3.2 Query 2: Identificar que estafetas entregaram determinada(s) encomenda(s) a um determinado cliente

Começamos por agrupar a lista de todos os estafetas, sem restrições.

Tendo esta obtida, verificamos para cada estafeta a sua lista de pedidos, de modo a verificar se realizaram alguma entrega ao cliente a cuja query se refere. Em caso positivo, este estafeta será guardado numa lista que, no final, será a nossa solução.

2.3.3 Query 3: Identificar os clientes servidos por um determinado estafeta

Neste caso, sabendo à posteriori o ID do estafeta, não necessitamos usar o findall, podendo portanto apenas anunciar o estafeta com os dados que pretendemos. Feito isto, obtemos a sua lista de pedidos, com a qual, guardamos de cada pedido o cliente servido, numa lista que será a solução do problema.

2.3.4 Query 4: Calcular o valor faturado pela Green Distribution num determinado dia

Como para algumas das queries anteriormente descritas, necessitamos da lista de todos os estafetas e, obtida esta, processemos cada estafeta, verificando os pedidos por ele respondido e, para cada, utilizamos uma função auxiliar que determina o preço da entrega. Este cálculo terá em conta o peso da encomenda e o prazo máximo que foi dado para a sua entrega, sendo que o preço aumentará conforme o aumento do peso ou de quão pequeno o intervalo de entrega se trata.

Calculado o preço, este será adicionado num somatório que será a solução desta questão.

2.3.5 Query 5: Identificar quais as zonas com maior volume de entregas por parte da Green Distribution

Explorando cada estafeta, a sua zona de trabalho será guardada, caso ainda não exista, numa lista, como um par zona/Acc, em que este acumulador, será o número de encomendas nessa específica zona, e, sempre que uma encomenda corresponder a esse local, o acumulador da lista será incrementado.

No final da procura da lista dos estafetas, faz-se uma procura do máximo acumulador na lista de pares zona/Acc, a partir do acumulado, guardando, no final, esse máximo como a solução.

2.3.6 Query 6: Calcular a classificação média de satisfação de cliente para um determinado estafeta

Tendo como base um ID de estafeta, basta utilizar o poder de busca do prolog para encontrar o estafeta em questão e, de seguida, calcular a sua classificação média dividindo o seu somatório das classificações e o número total de classificações.

2.3.7 Query 7: Identificar o número total de entregas pelos diferentes meios de transporte, num determinado intervalo de tempo

Calculada a lista de todos os estafetas, utilizamos como variáveis, três inteiros iniciados a zero, que, cada um correspondente a um tipo de veículo, serão incrementados, conforme o meio utilizado pelo estafeta e, a validade da data de entrega do pedido.

Para cada estafeta, será analisada a sua lista de pedidos, tendo sempre em conta o meio de transporte de eleição do estafeta. Para cada pedido, a data de entrega do mesmo será validada através de uma função auxiliar que recebe a data inferior do intervalo de tempo, a data superior e a data de entrega do pedido. No caso da data ser validada, a variável correspondente ao meio de transporte do estafeta é incrementada.

Concluída a análise da lista dos estafetas, toma-se por solução o valor das variáveis que foram sendo atualizadas durante a pesquisa.

2.3.8 Query 8: Identificar o número total de entregas pelos estafetas, num determinado intervalo de tempo

Semelhante à alínea anterior, para cada pedido será validada a data de entrega porém, em vez de se incrementar três variáveis, apenas queremos utilizar uma, sendo que, reutilizando o predicado auxiliar da query 7, entre cada estafeta, em vez de ser passadas três variáveis, apenas será passado o somatório delas com o calculado anteriormente.

Esta variável será o número de encomendas entregues num determinado intervalo de tempo.

2.3.9 Query 9: Calcular o número de encomendas entregues e não entregues pela Green Distribution, num determinado período de tempo

Na mesma lógica das últimas duas soluções, cada pedido será validado de acordo com um dado intervalo de tempo porém, novamente, as variáveis a ser atualizadas serão diferentes, sendo que

desta vez, teremos em conta a parcela de estado de um pedido, que nos indica se este foi ou não entregue.

Para cada estado do pedido, completo ou não, corresponderá uma variável do par solução, sendo que consoante o este estado, durante a pesquisa, a variável, deste par, a ele associado, será atualizada.

No final da pesquisa, teremos neste par, a resposta que pretendíamos, sendo a primeira parcela o número total de encomendas não entregues e a segunda, o número total de encomendas entregues.

2.3.10 Query 10: Calcular o peso total transportado por estafeta num determinado dia

Inicialmente acede-se à lista de estafetas por sua vez é percorrida e para cada lista de pedidos do estafeta calcula-se o peso total entregue para uma dada data.Esse calculo é feito por um filtro que percorre a lista e vai somando num acumulador o peso total dos pedidos que vão sendo filtrados.

2.3.11 Query 11 [EXTRA]: Calcular o peso total transportado por um dado estafeta num determinado dia

Dado um id de estafeta acede-se à lista de pedidos desse estafeta e calcula-se o peso total para uma dada data.Feito de forma análoga à query 10.

2.3.12 Query 12 [EXTRA]: Estafeta que mais entregou a um dado cliente

Em primeiro lugar, acede-se à lista de estafetas e em cada estafeta a lista de pedidos associada é percorrida e é contabilizado o número de entregas feitas para o cliente dado.O maior número contabilizado até ao momento é guardado até que no fim é obtido o estafeta que fez o maior número de entregas.

2.4 Adição e remoção de conhecimento

A inserção e remoção de conhecimento é efetuada à custa de invariantes que especificam um conjunto de restrições que devem ser verdadeiras após uma inserção/remoção de conhecimento.

A inserção de predicados pode ser vista como uma evolução do sistema em termos de conhecimento. Do mesmo modo, uma remoção de um predicado pode ser vista como uma involução do sistema.

De notar que, tanto as operações de inserção como de remoção são sempre efetuadas. No entanto, após cada uma dessas operações, é verificada a consistência do sistema com recurso aos invariantes.

Assim, utilizou-se maioritariamente 2 predicados nativos do PROLOG: `findall` e `length`. O primeiro encontra as ocorrências de X que satisfaçam Y e guarda-as numa lista L - invoca-se da seguinte forma: `findall(X,Y,L)`. Já o segundo calcula o comprimento de uma lista L, guardando o resultado em R - invoca-se da seguinte forma: `length(L,R)`.

2.4.1 Evolução do sistema

A inserção de predicados pode ser vista como uma **evolução** do sistema em termos de conhecimento. Assim sendo, este processo é traduzido da seguinte forma em PROLOG:

```
evolucao(T) :- findall(I,+T::I,Li),
               add_bc(T),
               teste(Li).

evolucao_troca(TA,TN) :- retract(TA),
                         evolucao(TN).
```

O `evolucao_troca` é um predicado que retira primeiro da base de conhecimento um termo antigo `TA`, antes de colocar um nome. Isso é-nos útil para caso queiramos atualizar os dados de um estafeta, visto que não podemos inserir na base de conhecimento simplesmente o estafeta atualizado, pois este terá o mesmo id do anterior e será "barrado"no invariante.

```
add_bc(T) :- assert(T).
add_bc(T) :- retract(T),!, fail.

teste([]).
teste([H|T]) :- H,
                teste(T).
```

O predicado `add_bc` é responsável por adicionar conhecimento ao sistema através do predicado nativo do PROLOG `assert`. Já o predicado `teste` serve para testar os invariantes dos termos inseridos na lista *Li*. Por conveniência, definimos que os invariantes da inserção de termos na base de conhecimento são precedidos pelo símbolo "+".

2.4.2 Involução do sistema

A remoção de predicados pode ser vista como uma **involução** do sistema. De notar que a implementação é muito semelhante à da inserção existindo, apenas, pequenas alterações.

```
involucao(T) :- findall(I,-T::I,Li),
                remove_bc(T),
                teste(Li).

remove_bc(T) :- retract(T),!, fail.
```

O predicado `remove_bc` é responsável por remover conhecimento ao sistema através do predicado nativo do PROLOG `retract`. Além disso, utilizamos o predicado `teste` com o mesmo intuito que na evolução do sistema. Por conveniência, definimos que os invariantes da remoção de termos na base de conhecimento são precedidos pelo símbolo -".

2.5 Invariantes

Como foi referido acima utilizamos invariantes para inserir ou remover entidades da base de conhecimento, estes invariantes são responsáveis por manter e integridade e a lógica do sistema que deve permanecer inalterada.

2.5.1 Estafeta

Inserção:

- Estafetas não podem ter o mesmo ID e este tem de ser um número (que é verificado através do predicado `integer`).

```
+estafeta(_,ID,_,_,_,_,_) :: (integer(ID),
                              findall(ID,estafeta(_,ID,_,_,_,_,_),S),
                              length(S,L),
                              L == 1).
```

- Estafeta com peso das encomendas menor ou igual ao peso máximo do seu transporte.

```
+estafeta(_,_,_,meio_transporte(_,_,_,P),_,LE,_) :: (pesoMenores(LE,P)).
```

- Estafeta possui um transporte único.

```
+estafeta(_,_,_,meio_transporte(ID,_,_,_),_,_,_) :: (integer(ID),
                    findall(ID,estafeta(_,_,_,meio_transporte(ID,_,_,_),_,_,_),S),
                    length(S,L),
                    L == 1).
```

- Estafeta só entrega na sua zona associada.

```
+estafeta(_,_,Z,_,_,LE,_) :: (morada(Z,_),
                              dentroZona(LE,Z)).
```

Remoção:

- Tem que existir pelo menos um estafeta na base de conhecimento.

```
-estafeta(Nome,ID,Z,MT,Cl,LE,P) :: (findall(estafeta(Nome,ID,Z,MT,Cl,LE,P),
                                            estafeta(Nome,ID,Z,MT,Cl,LE,P),S),
                                            length(S,L),
                                            L > 0).
```

2.5.2 Meio de transporte

Inserção:

- Dois meios de transporte não podem ter o mesmo id.

```
+meio_transporte(ID,_,_,_) ::(integer(ID),
                              findall(ID,meio_transporte(ID,_,_,_),S),
                              length(S,L), L == 1).
```

- Os meios de transporte tem que respeitar os tipos impostos e as restrições de pesos e velocidade máximas.

```
+meio_transporte(_,T,V,P) ::(transporte(T),
                             velMed(T,V),
                             pesoMax(T,P)
                             ).
```

- Tem que existir pelos menos um transporte na base de conhecimento.

```
-meio_transporte(ID,T,P,V) ::findall(meio_transporte(ID,T,P,V),
                              meio_transporte(ID,T,P,V),S),
                              length(S,L),
                              L > 0).
```

2.5.3 Pedido

Inserção

- Dois pedidos não podem ter o mesmo id.

```
+pedido(_,ID,_,_,_,_,_,_) ::(integer(ID),
                             findall(ID,pedido(_,ID,_,_,_,_,_,_),S),
                             length(S,L),
                             L == 1).
```

- Pedido tem que estar associado a uma morada correta.

```
+pedido(_,_,_,Rua,Zona,_,_,_) :: (morada(Zona,Rua)).
```

- A data de entrega tem que ser posterior à data do pedido.

```
+pedido(_,_,DataE,_,_,_,DataP,_) ::(valida_data(DataE),
                                    valida_data(DataP),
                                    data_valor(DataE,VE),
                                    data_valor(DataP,VP),
                                    VE > VP).
```

Remoção

- Tem que existir pelo menos um pedido na base de conhecimento.

```
-pedido(Cl,ID,DataE,R,Z,Pes,DataP,Est) ::(findall(pedido(Cl,ID,DataE,R,Z,Pes,DataP,Est),
                                          pedido(Cl,ID,DataE,R,Z,Pes,DataP,Est),S),
                                          length(S,L),
                                          L > 0).
```

2.5.4 Cliente

Inserção

- Dois clientes não podem ter o mesmo id e este tem de ser um número (que é verificado através do predicado `integer`).

```
+cliente(Nome,ID) ::(integer(ID),
                     findall(X,cliente(X,ID),S),
                     length(S,L),
                     (L == 1;
                     clienteUnico(S,Nome))).
```

Remoção

- Tem que existir pelo menos um cliente na base de conhecimento

```
-cliente(N,ID) ::(findall(cliente(N,ID),
                  cliente(N,ID),S),
                  length(S,L),
                  L > 0).
```

3 Conclusão

Depois de concluída a 1ª fase do trabalho prático desta UC, podemos observar que conseguimos implementar todas as *queries* básicas exigidas pela equipa docente com êxito, assim como algumas funcionalidades opcionais por iniciativa do grupo.

Fomos capazes de aplicar os conhecimentos lecionados durante as aulas práticas e reaproveitar exemplos de código apresentados que considerámos úteis, como por exemplo a `evolucao` e `involucao` do sistema.