Uma imagem com texto, ClipArt

Descrição gerada automaticamente

Universidade do Minho

**Relatório do Projeto Fase 1**

**Grupo 7**

Ano letivo 2021/2022

Dezembro 2021

Licenciatura em Engenharia Informática

Unidade Curricular de Inteligência Artificial

Uma imagem com pessoa, exterior

Descrição gerada automaticamenteUma imagem com exterior, árvore, pessoa, pose

Descrição gerada automaticamente

Ana Gonçalves a93259 Bruno Pereira a93298

Uma imagem com pessoa, parede, interior, jovem

Descrição gerada automaticamente

Francisco Toldy a93226 João Delgado a93240

Índice

[**1.** **Introdução** 3](#_Toc89121644)

[**2.** **Base de Conhecimento** 3](#_Toc89121645)

[**2.1.** 3](#_Toc89121646)

1. **Introdução**

O presente documento aborda o trabalho realizado na primeira fase do Instrumento de Avaliação em Grupo da cadeira de Inteligência Artificial. O mesmo começará por apresentar a interpretação do problema apresentado, indicando como este influenciou as escolhas tomadas sobre a base de conhecimento. De seguida apresentará a base de conhecimento e depois uma explicação elaborada da solução conseguida para cada uma das funcionalidades requeridas no enunciado fornecido.

1. **Base de Conhecimento**

O objetivo deste Instrumento de Avaliação era o 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, entre outros objetos". Para tal, foi necessário criar uma base de conhecimento para suportar os requisitos de cada uma das funcionalidades.

A base de conhecimentos assenta sobre 3 pilares

* 1. **Encomendas**

O predicado **encomenda/10** no qual é apresentada a informação sobre cada encomenda feita, tem 10 argumentos, sendo, por ordem, os seguintes:

* Id
* Peso
* Volume
* Classificação da encomenda (de 0 a 5)
* Rua
* Freguesia
* Veículo
* Preço
* Id do cliente
* Id do Estafeta

Esta informação foi considerada a apropriada tendo em conta as funcionalidades pedidas. Uma versão inicial da base de conhecimento possuía as datas importantes da encomenda, no entanto, foi tomada a decisão de separar as mesmas do predicado **encomenda/10**, dando origem a dois predicados diferentes



O predicado **dataE/4**, referente as entregas efetuadas, cujos argumentos são

* Id de encomenda (tal como no predicado anterior)
* Data da Encomenda
* Prazo da Encomenda
* Data de Entrega

No entanto, este predicado não refletia a necessidade de criar uma distinção entre as encomendas entregues e as que não foram entregues. Foi definido ainda então o predicado **dataNE/3** com os mesmos argumentos que o predicado anterior, mas sem o argumento relativo a data de Entrega.

****

****

* 1. **Clientes**

O segundo pilar da base de conhecimento é referente aos clientes e toda a informação importante para as funcionalidades que os referem. Foi definido o predicado **cliente/2** cujos argumentos são os seguintes

* Id do cliente
* Lista dos ids das encomendas que este cliente efetuou



* 1. **Estafeta**

O último pilar da base de conhecimentos são os estafetas, com o predicado **estafeta/5.** Cujos argumentos são os seguintes:

* Id do estafeta
* Número de encomendas entregues pelo mesmo
* *Rank* , ou seja, pontuação
* Lista dos ids das encomendas entregues
* Número de Penalizações

Uma versão inicial do predicado estafeta incluía a informação do veículo, no entanto, após reinterpretação do enunciado, foi tomada a decisão de mudar essa informação para o predicado Encomenda, uma vez que foi interpretado que cada encomenda de um mesmo estafeta poderia ser realizada recorrendo a diferentes veículos.



A informação relativa aos veículos é representada pelo predicado **veículo/2** que relaciona o tipo de veículo com uma pontuação ecológica (1 a 3) que é menor quão mais ecológico é o veículo.



1. **Tarefas**
   1. **Tarefa 1**
   2. **Tarefa 2: estafetasCliente/3**

O segundo requisito consiste em identificar que estafetas entregaram determinada(s) encomenda(s) a um determinado cliente. Com este objetivo criámos o predicado **estafetasCliente/3** ­­, este predicado recebe o identificador do cliente (*“****Cliente****”)* e uma lista de identificadores de encomendas **(“*[X|T]”)*,** destas quais, queremos saber o seu estafeta. No final, será devolvida uma lista de identificadores de estafetas (*“****[E|Result****]”*).

O predicado **estafetasCliente/3** consiste em 2 definições: o caso base e o cado recursivo.O programa entra no caso base quando a lista de encomendas sobre as quais desejamos saber informações está vazia e, por isso, a lista de estafetas será também vazia.



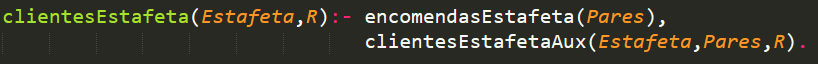
O caso recursivo, é efetivamente onde colocamos no resultado a lista de estafetas. Iniciamos com o id da primeira encomenda e encontramos o id do Estafeta, e esse estafeta será adicionado ao resultado que já foi calculado com o resto da lista de encomendas através da recursividade.

Uma imagem com texto, laranja

Descrição gerada automaticamente

* 1. **Tarefa 3: clientesEstafeta/3**

Este requisito de identificar os clientes servidos por um determinado estafeta, foi concretizado através do predicado **clientesEstafeta/3***.* Este predicado recebe o estafeta do qual queremos saber os seus clientes e devolverá uma lista de identificadores de clientes.

**

Com a utilização do predicado auxiliar **encomendasEstafeta/1**conseguimos obter numa lista os pares (Estafeta, Lista de Encomendas do Estafeta). Com o predicado **clientesEstafetaAux/3**obtemos o resultado desejado, através de 3 casos:

* O caso base, em que a lista de pares (Estafeta, Lista de Encomendas) encontra-se vazia, e por isso a lista de clientes a que as encomendas foram entregues é também vazia.



* No segundo caso, em que não encontramos na lista de pares o estafeta que nos foi requisitado, e por isso chamamos a recursiva para o encontrar no resto da lista.



* No último caso, em que encontramos o estafeta e então chamamos o predicado **clientesEstafetaAux2/2** que vai receber a lista de Encomendas e devolver a lista de Clientes.

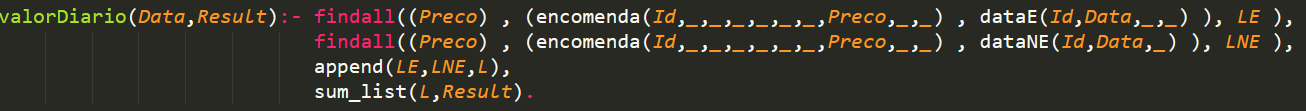


****

* 1. **Tarefa 4: valorDiario/2**

Para esta tarefa, é necessário calcular o valor total faturado pela empresa num determinado dia.

Aqui, foram usadas duas cláusulas **findall** para compilar numa lista os preços de todas as encomendas, entregues ou não entregues, cuja data seja a especificada como argumento. A razão por que são necessários ambos os **findall** é a separação que foi feita das datas de encomendas entregues e não entregues. Nos **findall**, vamos buscar à base de conhecimento todas as ocorrências de encomendas instanciadas com a data especificada, unificando os preços numa lista, tanto nas encomendas entregues como nas não entregues, LE e LNE, respetivamente. Uma vez que são geradas duas listas, é feito o **append** das mesmas e colocado o resultado em L. Por fim, é feito o somatório de todos os elementos da lista L e o valor final é colocado em Result. Assim, é calculado o valor total faturado num determinado dia.



* 1. **Tarefa 5: maisEntregasFreg/1**

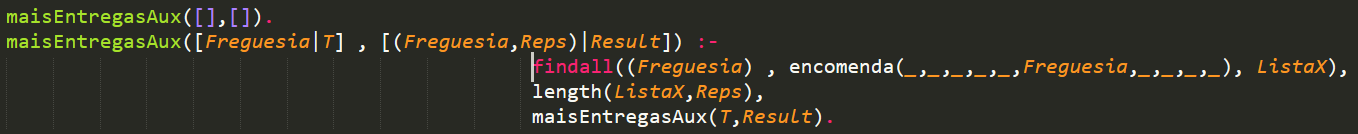
Identificar quais as zonas (e.g., rua ou freguesia) com maior volume de entregas por parte da *Green Distribution*;

Uma imagem com texto

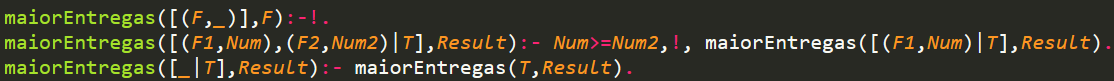
Descrição gerada automaticamente

Para esta **tarefa** será necessário, mais uma vez com um **findall**, compilar todas as freguesias numa lista Freguesias. Depois de coletadas as Freguesias, são desta lista retiradas todas as ocorrências repetidas. Depois, é chamada o predicado **maisEntregasAux/2**.

Este predicado, recebendo a lista de Freguesias existentes, irá gerar uma lista de pares no formato (Freguesia, Ocorrências). Para o fazer, será percorrida a lista de Freguesias, em que para cada elemento, é usado um **findall**, em que são unificadas todas as encomendas cuja freguesia seja a que está a ser analisada, e coloca a Freguesia numa lista, tantas vezes quantas o número de encomendas que a ela são relativas. Depois disso, é calculada o tamanho desta lista, ou seja, por outras palavras, o número de encomendas que foram efetuadas com destino a esta Freguesia. Estes pares são colocados na lista resultado.

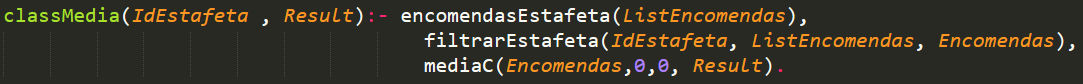


Finalmente, será chamado o predicado **maiorEntregas**. Este, por sua vez, pegará na lista de pares gerada pela **maisEntregasAux** e calculará a freguesia com mais ocorrências. Para isto é usado um predicado em que são, sucessivamente, comparados os dois primeiros elementos, para ver qual deles tem mais ocorrências e é removido da lista o que tiver menos. No caso de paragem, quando estiver na lista apenas um elemento, este será o par que conterá a Freguesia com maior número de encomendas. Este resultado será colocado na variável Result.

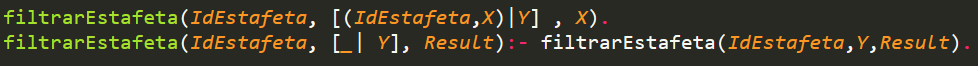


* 1. **Tarefa 6: classMedia/2**

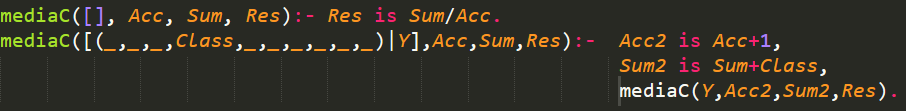
Esta funcionalidade tem como objetivo cálculo da classificação média de um dado Estafeta. Para tal foi criado o predicado **classMedia/2** que começa por criar uma lista de pares (X,Y) em que X é o Estafeta e Y a Lista de Encomendas entregues pelo estafeta.



Dada essa lista de pares, o predicado **filtrarEstafeta/3** , filtra a lista formada pelo predicado anterior, ficando em ***Encomendas*** apenas a lista de encomendas entregues pelo Estafeta em questão.



Por último, o predicado **mediaC/4** calcula a classificação media do estafeta, efetuando uma média das classificações das encomendas na lista criada pelo predicado **filtrarEstafeta/3.**

****

* 1. **Tarefa 7: entregasPeriodoTransporte/3**

Para esta query, é necessário identificar, para cada meio de transporte, o número total de entregas que foram feitas, num determinado intervalo de tempo.

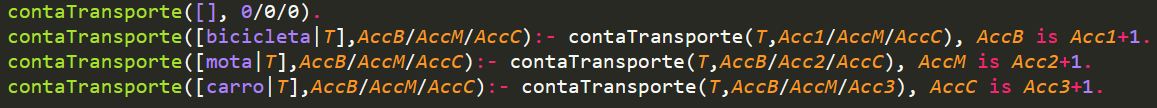
Para atingir este objetivo, recorremos ao predicado **entregasPeriodoTransporte/3**, onde foram colocados em L todos os transportes das encomendas cuja data estivesse dentro da data especificada, com o auxílio de um **findall**, novamente.

Para a comparação das datas, foram criados dois predicados **antes/2** e **depois/2**, que determinam se uma determinada data está antes ou depois de uma outra data, respetivamente.

Uma imagem com texto

Descrição gerada automaticamente

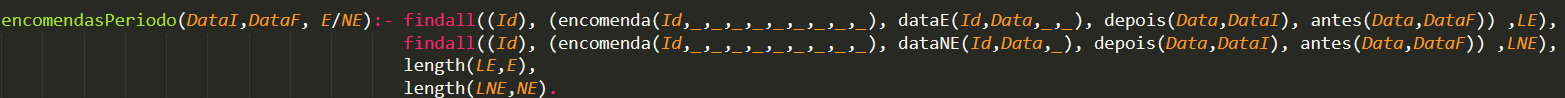
Depois de compilados todos os transportes numa lista, resta percorrê-la e fazer a contagem das ocorrências de cada meio de transporte. Para este fim, foi escrito o predicado **contaTransporte**.



Este predicado recebe a lista de transportes gerada no predicado principal e analisa os seus elementos, gerando no final um argumento do tipo OcorrênciasBicicleta/OcorrênciasMota/OcorrênciasCarro. Analisando a cabeça da lista, caso esta seja bicicleta, o valor OcorrênciasBicicleta será incrementado, sendo o processo semelhante para os restantes meios de transporte nas iterações seguintes. Assim, será devolvido o resultado neste formato.

* 1. **Tarefa 8:**
  2. **Tarefa 9: encomendasPeriodo/3**

Para calcular o número de encomendas entregues e não entregues pela *Green Distribution*, num determinado período de tempo, criamos o predicado **encomendasPeriodo/3.** Este receberá em primeiro lugar a data inicial e a data final que queremos filtrar, e vai devolver um par com o número de encomendas entregues e não entregues.



Na primeira condição estamos a reunir todos os identificadores das encomendas que se encontram no período de tempo pedido na lista LE. O mesmo estamos a fazer para as encomendas não entregues, na lista LNE. Finalmente, colocamos no par o número de identificadores tanto das entregues, como das não entregues no valor a devolver.

* 1. **Tarefa 10:**

1. **Conclusões**